Titre: Title:	On High-Performance Benders-Decomposition-Based Exact Methods with Application to Mixed-Integer and Stochastic Problems
Auteur: Author:	Ragheb Rahmaniani
Date:	2018
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Rahmaniani, R. (2018). On High-Performance Benders-Decomposition-Based Exact Methods with Application to Mixed-Integer and Stochastic Problems [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3008/

UNIVERSITÉ DE MONTRÉAL

ON HIGH-PERFORMANCE BENDERS-DECOMPOSITION-BASED EXACT METHODS WITH APPLICATION TO MIXED-INTEGER AND STOCHASTIC PROBLEMS

RAGHEB RAHMANIANI
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INDUSTRIEL)
MARS 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

ON HIGH-PERFORMANCE BENDERS-DECOMPOSITION-BASED EXACT METHODS WITH APPLICATION TO MIXED-INTEGER AND STOCHASTIC PROBLEMS

présentée par : RAHMANIANI Ragheb
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. EL HALLAOUI Issmaïl, Ph. D., président
M. GENDREAU Michel, Ph. D., membre et directeur de recherche
M. CRAINIC Teodor G., Ph. D., membre et codirecteur de recherche
M. REI Walter, Ph. D., membre et codirecteur de recherche
M. GENDRON Bernard, Ph. D., membre
M. WONG Richard, Ph. D., membre externe

DEDICATION

To my parents and family ...

ACKNOWLEDGMENTS

First and foremost, I wholeheartedly appreciate my wonderful advisors, Profs. Michel Gendreau, Teodor Gabriel Crainic and Walter Rei for their excellent guidance, immense knowledge and motivations. They supported me very patiently by giving me the opportunity to explore many different research areas during my PhD. I could not have imagined more supportive, passionate, dedicated, and considerate advisors for my PhD Study. I consider them as role models in my future careers. I can only hope to be a problem-solver and a teacher as well as them.

I cannot ever thank Prof. Shabbir Ahmed enough for hosting me as a visiting student at H. Milton Stewart School of Industrial and Systems Engineering (ISyE) at Georgia Institute of Technology. It was really a unique opportunity to collaborate with Prof. Ahmed and be a part of his research group. He was and will always be a great inspiration to me.

I would like to specially thank Profs. Ismail El Hallaoui, Bernard Gendron and Richard Wong for accepting to serve on my thesis committee. I really appreciate their attention and feedbacks.

Furthermore, I acknowledge The Fonds de recherche du Québec - Nature et technologies (FRQNT) for providing a full financial scholarships during the fourth year of my PhD study that supported my visit at ISyE. Likewise, I express my gratitude to CIRRELT for supporting my research by several scholarships.

I would like to extend my gratitudes to all my friends at CIRRELT and GERAD for their endless help, support, and motivation. I was very fortunate to start my researches in such a warm and dynamic environment. I have spent many hours in productive and fruitful conversation on research and life with my outstanding officemates. In addition, I would like to acknowledge with much appreciation the staff at CIRRELT and ISyE. I was very much impressed by their dedication and preciseness in handling the bureaucratic procedures. I would like to give my special thanks to Serge Bisaillon for helping me with the codings.

RÉSUMÉ

La programmation stochastique en nombres entiers (SIP) combine la difficulté de l'incertitude et de la non-convexité et constitue une catégorie de problèmes extrêmement difficiles à résoudre. La résolution efficace des problèmes SIP est d'une grande importance en raison de leur vaste applicabilité. Par conséquent, l'intérêt principal de cette dissertation porte sur les méthodes de résolution pour les SIP. Nous considérons les SIP en deux étapes et présentons plusieurs algorithmes de décomposition améliorés pour les résoudre. Notre objectif principal est de développer de nouveaux schémas de décomposition et plusieurs techniques pour améliorer les méthodes de décomposition classiques, pouvant conduire à résoudre optimalement divers problèmes SIP.

Dans le premier essai de cette thèse, nous présentons une revue de littérature actualisée sur l'algorithme de décomposition de Benders. Nous fournissons une taxonomie des améliorations algorithmiques et des stratégies d'accélération de cet algorithme pour synthétiser la littérature et pour identifier les lacunes, les tendances et les directions de recherche potentielles. En outre, nous discutons de l'utilisation de la décomposition de Benders pour développer une (méta-)heuristique efficace, décrire les limites de l'algorithme classique et présenter des extensions permettant son application à un plus large éventail de problèmes.

Ensuite, nous développons diverses techniques pour surmonter plusieurs des principaux inconvénients de l'algorithme de décomposition de Benders. Nous proposons l'utilisation de plans de coupe, de décomposition partielle, d'heuristiques, de coupes plus fortes, de réductions et de stratégies de démarrage à chaud pour pallier les difficultés numériques dues aux instabilités, aux inefficacités primales, aux faibles coupes d'optimalité ou de réalisabilité, et à la faible relaxation linéaire. Nous testons les stratégies proposées sur des instances de référence de problèmes de conception de réseau stochastique. Des expériences numériques illustrent l'efficacité des techniques proposées.

Dans le troisième essai de cette thèse, nous proposons une nouvelle approche de décomposition appelée méthode de décomposition primale-duale. Le développement de cette méthode est fondé sur une reformulation spécifique des sous-problèmes de Benders, où des copies locales des variables maîtresses sont introduites, puis relâchées dans la fonction objective. Nous montrons que la méthode proposée atténue significativement les inefficacités primales et duales de la méthode de décomposition de Benders et qu'elle est étroitement liée à la méthode de décomposition duale lagrangienne. Les résultats de calcul sur divers problèmes SIP montrent la supériorité de cette méthode par rapport aux méthodes classiques de décomposition.

Enfin, nous étudions la parallélisation de la méthode de décomposition de Benders pour étendre ses performances numériques à des instances plus larges des problèmes SIP. Les variantes parallèles disponibles de cette méthode appliquent une synchronisation rigide entre les processeurs maître et esclave. De ce fait, elles souffrent d'un important déséquilibre de charge lorsqu'elles sont appliquées aux problèmes SIP. Cela est dû à un problème maître difficile qui provoque un important déséquilibre entre processeur et charge de travail. Nous proposons une méthode Benders parallèle asynchrone dans un cadre de type branche-et-coupe. L'assouplissement des exigences de synchronisation entraine des problèmes de convergence et d'efficacité divers auxquels nous répondons en introduisant plusieurs techniques d'accélération et de recherche. Les résultats indiquent que notre algorithme atteint des taux d'accélération plus élevés que les méthodes synchronisées conventionnelles et qu'il est plus rapide de plusieurs ordres de grandeur que CPLEX 12.7.

ABSTRACT

Stochastic integer programming (SIP) combines the difficulty of uncertainty and non-convexity, and constitutes a class of extremely challenging problems to solve. Efficiently solving SIP problems is of high importance due to their vast applicability. Therefore, the primary focus of this dissertation is on solution methods for SIPs. We consider two-stage SIPs and present several enhanced decomposition algorithms for solving them. Our main goal is to develop new decomposition schemes and several acceleration techniques to enhance the classical decomposition methods, which can lead to efficiently solving various SIP problems to optimality.

In the first essay of this dissertation, we present a state-of-the-art survey of the Benders decomposition algorithm. We provide a taxonomy of the algorithmic enhancements and the acceleration strategies of this algorithm to synthesize the literature, and to identify shortcomings, trends and potential research directions. In addition, we discuss the use of Benders decomposition to develop efficient (meta-)heuristics, describe the limitations of the classical algorithm, and present extensions enabling its application to a broader range of problems.

Next, we develop various techniques to overcome some of the main shortfalls of the Benders decomposition algorithm. We propose the use of cutting planes, partial decomposition, heuristics, stronger cuts, and warm-start strategies to alleviate the numerical challenges arising from instabilities, primal inefficiencies, weak optimality/feasibility cuts, and weak linear relaxation. We test the proposed strategies with benchmark instances from stochastic network design problems. Numerical experiments illustrate the computational efficiency of the proposed techniques.

In the third essay of this dissertation, we propose a new and high-performance decomposition approach, called Benders dual decomposition method. The development of this method is based on a specific reformulation of the Benders subproblems, where local copies of the master variables are introduced and then priced out into the objective function. We show that the proposed method significantly alleviates the primal and dual shortfalls of the Benders decomposition method and it is closely related to the Lagrangian dual decomposition method. Computational results on various SIP problems show the superiority of this method compared to the classical decomposition methods as well as CPLEX 12.7.

Finally, we study parallelization of the Benders decomposition method. The available parallel variants of this method implement a rigid synchronization among the master and slave

processors. Thus, it suffers from significant load imbalance when applied to the SIP problems. This is mainly due to having a hard mixed-integer master problem that can take hours to be optimized. We thus propose an asynchronous parallel Benders method in a branch-and-cut framework. However, relaxing the synchronization requirements entails convergence and various efficiency problems which we address them by introducing several acceleration techniques and search strategies. In particular, we propose the use of artificial subproblems, cut generation, cut aggregation, cut management, and cut propagation. The results indicate that our algorithm reaches higher speedup rates compared to the conventional synchronized methods and it is several orders of magnitude faster than CPLEX 12.7.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Mixed-integer programming and *stochastic programming* are essential in addressing some of the most important engineering and financial applications of optimization, e.g., energy (Wallace and Fleten, 2003), finance (yong Yu et al., 2003; Simsek, 2008), healthcare (Rais and Viana, 2011), security (Pan and Morton, 2008), transportation (Crainic and Laporte, 1997; Rahmaniani and Ghaderi, 2013), manufacturing (Zanjani et al., 2013), supply chain (Santoso et al., 2005), telecommunication (Gendron et al., 1999), just to name a few. These two classes of the optimization problems are known as the more challenging classes of the mathematical programming (Nemhauser and Wolsey, 1988; Birge and Louveaux, 1997; Jünger et al., 2009; Küçükyavuz and Sen, 2017). It is then no surprise that their combination, i.e., *stochastic integer programming* (SIP), endures serious conceptual and computational challenges.

Solving realistic SIP problems with off-the-shelf optimization solvers such as CPLEX and Gurobi is disappointing, see, e.g., the results in Chapters 4 to 6. This is due to the non-convexities as well as the large number of scenarios required to properly set the value of the uncertain parameters in the model (Kall et al., 1994; Küçükyavuz and Sen, 2017). For this reason, *decomposition* methods are essential and have become the methodology of choice when dealing with the realistic instances of the SIP problems (Ruszczyński, 1997; Ruszczyński, 2003).

The stochastic problems often exhibit a so-called *dual decomposition structure* and the dual of its LP relaxation has a *block angular structure* (Van Slyke and Wets, 1969). Successful algorithms take advantage of this special structure to decompose the problem into smaller pieces, yielding the well-known and classical *Benders decomposition* (BD) algorithm (Benders, 1962) or *Lagrangian dual decomposition* (LDD) method (Carøe and Schultz, 1999).

These decomposition methods solve successively several *subproblems* and a *master problem* until an optimal solution is achieved. The BD subproblem is a restriction of the original problem wherein a specific set of the variables are fixed. At each iteration, the value of these variables is adjusted by solving the master problem. In the LDD method, the subproblems are obtained by applying Lagrangian relaxation to some (complicating) constraints. Each iteration of this method consists of selecting a new set of Lagrangian multipliers by solving the master problem and solving the subproblems for the chosen multipliers. The master problems are adjusted using the solution obtained from solving the subproblems, e.g., adding cuts.

These methods have been subject of intense research due to their practical importance in

handling a wide gamut of challenging SIP problems (Ruszczyński, 1999, 2003; Wallace and Ziemba, 2005; Uryasev and Pardalos, 2013). However, the computational performance of these decomposition methods usually depends on some (problem-specific) acceleration techniques (Jojic et al., 2010; Rush and Collins, 2012; Rahmaniani et al., 2017a). For instance, the BD method usually becomes ineffective if the underlying LP relaxation is weak and the LDD method often exhibits slow convergence due to having hard-to-solve subproblems, updating the multipliers, and difficulty in generating good feasible solutions from the subproblems. Likewise, one of the most time consuming part of these decomposition methods lies in solving the subproblems at each iteration. This is because a large number of scenarios are usually required to properly set the value of the uncertain parameters and the decomposition methods yield a subproblem for each scenario. These scenario-subproblems are disjoint. Therefore, in addition to mathematical techniques, parallel computing techniques are oftentimes used to accelerate the numerical performance of these methods (Nielsen and Zenios, 1997; Linderoth and Wright, 2003; Crainic, 2015; Deng et al., 2018).

While many efforts have been made to accelerate the computational performance of these decomposition methods, the existing methods are yet far from introducing a high-performing algorithm. In fact, as the results of this dissertation illustrate, there is still significant room for improvements. Thus, in this dissertation, we strive to improve the efficiency of decomposition algorithms, particularly the BD method.

This dissertation consists of four papers, concerning the synthesis of the literature, development of the methodological techniques to accelerate the BD method, and presenting a new decomposition scheme which is closely related to the BD and the LDD methods. Most of our developments are general and applicable to the two main branches of mathematical optimization, i.e., mixed-integer and stochastic programming.

To analyze the computational performance of our developments, we consider a wide range of benchmark instances related to various important and challenging optimization problems. In particular, we consider *network design* problems (Magnanti and Wong, 1984; Crainic and Laporte, 1997; Gendron et al., 1999; Klibi et al., 2010) as one of our main applications in assessing our methods. This family of challenging problems naturally appear in many applications (Klibi et al., 2010) and has served as one of the most attractive testbeds for the BD method (Costa, 2005).

We close this section with an overview of the contributions and structure of this dissertation, which provides various numerical and theoretical contributions to two main branches of mathematical optimization, i.e., mixed-integer and stochastic programming.

1.1 Thesis contribution

The most notable contributions of this dissertation are as follows :

- In the first essay,
  - Presenting the first comprehensive literature review of the BD method ;
- In the second essay,
  - Proposing new valid inequalities for the network design problems when the BD method is used to solve them ;
  - Proposing the use of add-on cutting plane methods to iteratively add valid inequalities to the master and sub- problems of the BD method which significantly reduces the number of the iterations and tightens the LP relaxation ;
  - Revisiting the generation of Pareto-optimal cuts to accelerate the resolution of the auxiliary subproblem ;
  - Proposing an alternative strategy to the classical feasibility cuts by making use of auxiliary variables and strengthened combinatorial cuts ;
  - Proposing a simple and effective warm-start strategy to quickly generate Benders cuts ;
  - Proposing a state-of-the-art Benders decomposition algorithm ;
- In the third essay,
  - Proposing a new decomposition technique which combines the complementary advantages of the classical decomposition methods, i.e., Benders and Lagrangian dual decomposition method ;
  - Showing that the algorithm can theoretically generate stronger optimality and feasibility cuts than the BD method ;
  - Demonstrating that our method can converge to an optimal integer solution at the root node, i.e., no branching effort being required ;
  - Developing various numerically efficient implementation methodologies for the proposed decomposition strategy ;
- In the fourth essay,
  - Presenting an extensive literature review and synthesize of the existing parallel BD methods ;
  - Presenting parallelization strategies for the BD algorithm in a branch-and-cut framework ;
  - Designing a framework which maintains the global convergence of our parallel algorithms when asynchronous communications are used ;
  - Modifying some the classical acceleration strategies to properly fit into our parallel

algorithms ;
- Proposing new acceleration strategies to increase the efficiency of our parallel algorithms.

1.2 Thesis organization

Chapter 2, the first essay of this dissertation, reviews and synthesizes the decades of research on the BD method in order to clarify the state-of-the-art techniques of the BD's toolbox and to identify shortcomings, trends, potential research directions, its relation to other methods and its extensions.

In Chapter 3, we recall some of the most important definitions and concepts that we extensively use in this dissertation. This chapter helps the readers to have a better insight on the methodologies that we develop in this dissertation.

Chapter 4 constitutes the second essay of this dissertation. This essay presents various empirical and mathematical techniques to overcome some of the main numerical inefficiencies of the BD method in solving mixed-integer problems, particularly SIPs. Specifically, to obtain a state of the art Benders method, it presents various acceleration techniques, including the use of cutting planes, partial decomposition, heuristics, stronger cuts, reduction and warm-start strategies. This chapter finishes with extensive computational experiments conducted on benchmark instances and compares the obtained results to the existing methods.

The third essay of this dissertation is presented in Chapter 5. In this chapter a novel decomposition algorithm is proposed which is closely related to the Benders and Lagrangian dual decomposition methods. The proposed method is illustrated over a toy example and various implementation strategies are proposed. This chapter concludes with the presentation of the numerical comparisons versus the classical decomposition methods and CPLEX 12.7.

Chapter 6, forming the fourth essay of this dissertation, addresses the parallelization techniques for the BD method in a branch-and-cut framework. The existing parallel BD algorithms are reviewed and the relation of our algorithm to other parallel branch-and-cut is illustrated. Moreover, to enhance the proposed algorithms various acceleration techniques and search strategies are proposed. This chapter concludes with the presentation of the numerical results and an evaluation of the performance of our parallel algorithms.

Finally, Chapter 7 provides summary of this dissertation and Chapter 8 gives the conclusions, including the limitations of our work and various fruitful future research directions.

CHAPTER 2 CRITICAL LITERATURE REVIEW: ARTICLE 1: THE BENDERS DECOMPOSITION ALGORITHM: A LITERATURE REVIEW

Ragheb Rahmaniani[1], Teodor Gabriel Crainic[2], Michel Gendreau[1], Walter Rei[2]

[1] CIRRELT & Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal H3C 3A7, Canada.

[2] CIRRELT & School of Management, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal H3C 3P8, Canada.

Abstract. The Benders decomposition algorithm has been successfully applied to a wide range of difficult optimization problems. This paper presents a state-of-the-art survey of this algorithm, emphasizing its use in combinatorial optimization. We discuss the classical algorithm, the impact of the problem formulation on its convergence, and the relationship to other decomposition methods. We introduce a taxonomy of algorithmic enhancements and acceleration strategies based on the main components of the algorithm. The taxonomy provides the framework to synthesize the literature, and to identify shortcomings, trends and potential research directions. We also discuss the use of the Benders Decomposition to develop efficient (meta-)heuristics, describe the limitations of the classical algorithm, and present extensions enabling its application to a broader range of problems.

Keywords. *Combinatorial optimization*, *Benders decomposition*, *Acceleration techniques*, *Literature review.*

History. This article is published in European Journal of Operational Research.

2.1 Introduction

It has been more than five decades since the *Benders Decomposition* (*BD*) algorithm was proposed by Benders (1962), with the main objective of tackling problems with *complicating variables,* which, when temporarily fixed, yield a problem significantly easier to handle. The BD method (also referred to as *variable partitioning* (Zaourar and Malick, 2014) and *outer linearization* (Trukhanov et al., 2010)) has become one of the most widely used exact algorithms, because it exploits the structure of the problem and decentralizes the overall computational burden. Successful applications are found in many divers fields, including planning and scheduling (Hooker, 2007; Canto, 2008), health care (Luong, 2015), transportation and telecommunications (Costa, 2005), energy and resource management (Cai et al., 2001; Zhang and Ponnambalam, 2006), and chemical process design (Zhu and Kuno, 2003), as illustrated

in Table 2.1.

Table 2.1 Some applications of the Benders decomposition method

	Reference	Application		Reference	Application
1	Behnamian (2014)	Production planning	17	Jiang et al. (2009)	Distribution planning
2	Adulyasak et al. (2015)	Production routing	18	Wheatley et al. (2015)	Inventory control
3	Boland et al. (2015)	Facility location	19	Laporte et al. (1994)	Traveling salesman
4	Boschetti and Maniezzo (2009)	Project scheduling	20	Luong (2015)	Healthcare planning
5	Botton et al. (2013)	Survivable network design	21	Maravelias and Grossmann (2004)	Chemical process design
6	Cai et al. (2001)	Water resource management	22	Moreno-Centeno and Karp (2013)	Implicit hitting sets
7	Canto (2008)	Maintenance scheduling	23	Oliveira et al. (2014)	Investment planning
8	Codato and Fischetti (2006)	Map labeling	24	Osman and Baki (2014)	Transfer line balancing
9	Cordeau et al. (2006)	Logistics network design	25	Pérez-Galarce et al. (2014)	Spanning tree
10	Cordeau et al. (2001a)	Locomotive assignment	26	Pishvaee et al. (2014)	Supply chain network design
11	Cordeau et al. (2001b)	Airline scheduling	27	Rubiales et al. (2013)	Hydrothermal coordination
12	Corréa et al. (2007)	Vehicle routing	28	Saharidis et al. (2011)	Refinery system network planning
13	Côté et al. (2014)	Strip packing	29	Sen et al. (2015)	Segment allocation
14	Fortz and Poss (2009)	Network design	30	Bloom (1983)	Capacity expansion
15	Gelareh et al. (2015)	Transportation	31	Wang et al. (2016a)	Optimal power flow
16	Jenabi et al. (2015)	Power management	32	Errico et al. (2016)	Public transit

The BD method is based on a sequence of projection, outer linearization, and relaxation (Geoffrion, 1970a,b). Thus, the model is first projected onto the subspace defined by the set of complicating variables. The resulting formulation is then dualized, and the associated extreme rays and points respectively define the feasibility requirements (feasibility cuts) and the projected costs (optimality cuts) of the complicating variables. Thus, an equivalent formulation can be built by enumerating all the extreme points and rays. However, performing this enumeration and, then, solving the resulting formulation is generally computationally exhausting, if not impossible. Hence, one solves the equivalent model by applying a relaxation strategy to the feasibility and optimality cuts, yielding a *Master Problem* (*MP*) and a subproblem, which are iteratively solved to respectively guide the search process and generate the violated cuts.

The BD algorithm was initially proposed for a class of mixed-integer linear programming (*MILP*) problems. When the integer variables are fixed, the resulting problem is a continuous linear program (*LP*) for which we can use standard duality theory to develop cuts. Many extensions have since been developed to apply the algorithm to a broader range of problems (e.g., Geoffrion, 1972; Hooker and Ottosson, 2003). Other developments were proposed to increase the algorithm's efficiency on certain optimization classes (e.g., Costa et al., 2012; Crainic et al., 2014a). In addition, BD often provides a basis for the design of effective heuristics for problems that would otherwise be intractable (Côté and Laughton, 1984; Raidl, 2015). The BD approach has thus become widely used for linear, nonlinear, integer, stochastic, multi-stage, bilevel, and other optimization problems, as illustrated in Table 2.2.

Figure 2.1 depicts the increasing interest in the BD algorithm over the years. Despite this level

Table 2.2 Examples of optimization problems handled via Benders method

	Reference	Model		Reference	Model
1	Adulyasak et al. (2015)	Multi-period stochastic problem	16	Jenabi et al. (2015)	Piecewise linear mixed-integer problem
2	Behnamian (2014)	Multi-objective MILP	17	Wolf (2014)	Multi-stage stochastic program
3	Cai et al. (2001)	Multi-objective nonconvex nonlinear problem	18	Laporte et al. (1994)	Probabilistic integer formulation
5	Cordeau et al. (2001b)	Pure 0–1 formulation	19	Li (2013)	Large-scale nonconvex MINLP
6	Corréa et al. (2007)	Binary problem with logical expressions	20	Moreno-Centeno and Karp (2013)	Problem with constraints unknown in advance
7	Gabrel et al. (1999)	Step increasing cost	21	Bloom (1983)	Nonlinear multi-period problem with reliability constraint
8	Côté et al. (2014)	MILP with logical constraints	22	Osman and Baki (2014)	Nonlinear integer formulation
9	de Camargo et al. (2011)	Mixed-integer nonlinear program (MINLP)	23	Pérez-Galarce et al. (2014)	Minmax regret problem
10	Emami et al. (2016)	Robust optimization problem	24	Pishvaee et al. (2014)	Multi-objective possibilistic programming model
11	Fontaine and Minner (2014)	Bi-level problem with bilinear constraints	25	Raidl et al. (2014)	Integer, bilevel, capacitated problem
12	Fortz and Poss (2009)	Multi-layer capacitated network problem	27	Rubiales et al. (2013)	Quadratic MILP master problem and nonlinear subproblem
13	Gendron et al. (2016)	Binary problem with nonlinear constraints	28	Sahinidis and Grossmann (1991)	MINLP and nonconvex problems
14	Grothey et al. (1999)	Convex nonlinear problem	29	Harjunkoski and Grossmann (2001)	Multi-stage problem with logical and big-M constraints
15	O'Kelly et al. (2014)	MINLP with concave objective function and staircase constraint matrix structure			

of interest, there has been no comprehensive survey of the method in terms of its numerical and theoretical challenges and opportunities. The now out-of-date survey by Costa (2005) reviews only applications to fixed-charge network design problems. The main goal of this paper therefore is to contribute to filling this gap by reviewing the current state-of-the-art, focusing on the main ideas for accelerating the method, discussing the main variants and extensions aimed to handle more general problems involving, e.g., nonlinear/integer/constraint programming subproblems, and identifying trends and promising research directions.

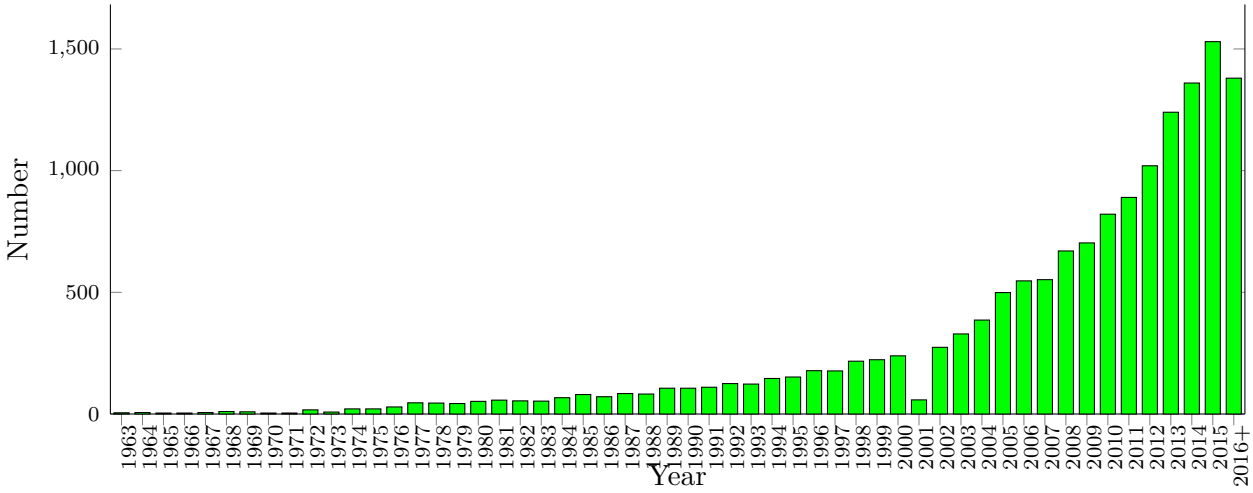

Figure 2.1 Annual number of mentions of the Benders decomposition according to https ://scholar.google.com/.

Many different enhancement strategies were proposed to address the shortcomings of the BD method and accelerate it. This effort contributed significantly to the success of the method. We propose a taxonomy of the enhancement and acceleration strategies based on the main components of the algorithm : the decomposition strategy, the strategies to handle the MP

and subproblem, and the strategies to generate solutions and cuts. The taxonomy provides the framework to classify and synthesize the literature and to identify relations among strategies and between these and the BD method.

The remainder of this article is organized as follows. Section 2.2 presents the classical BD algorithm, the associated model selection criteria, and its relationship to other decomposition methods. Section 2.3 presents the proposed taxonomy, used to survey the acceleration strategies in Sections 2.4 to 2.7. Section 2.8 presents Benders-type heuristics, and Section 2.9 describes extensions of the classical algorithm. Finally, Section 2.10 provides concluding remarks and describes promising research directions.

2.2 The Benders decomposition method

We present in this section the classical version of the Benders algorithm (Benders, 1962). We review its extensions to a broader range of optimization problems in Section 2.9.

2.2.1 The classical version

We consider an MILP of the form

$$\begin{aligned} \text{Minimize} \quad & f^T y + c^T x && (2.1)\\ \text{subject to} \quad & Ay = b && (2.2)\\ & By + Dx = d && (2.3)\\ & x \geq 0 && (2.4)\\ & y \geq 0 \quad \text{and integer}, && (2.5) \end{aligned}$$

with complicating variables $y \in \mathbb{R}^{n_1}$, which must take positive integer values and satisfy the constraint set $Ay = b$, where $A \in \mathbb{R}^{m_1 \times n_1}$ is a known matrix and $b \in \mathbb{R}^{m_1}$ is a given vector. The continuous variables $x \in \mathbb{R}^{n_2}$, together with the y variables, must satisfy the linking constraint set $By + Dx = d$, with $B \in \mathbb{R}^{m_2 \times n_1}$, $D \in \mathbb{R}^{m_2 \times n_2}$, and $d \in \mathbb{R}^{m_2}$. The objective function minimizes the total cost with the cost vectors $f \in \mathbb{R}^{n_1}$ and $c \in \mathbb{R}^{n_2}$.

Model (2.1–2.5) can be re-expressed as

$$\min_{\bar{y} \in Y} \{ f^T \bar{y} + \min_{x \geq 0} \{ c^T x : Dx = d - B\bar{y} \}, \tag{2.6}$$

where $\bar{y}$ is a given value for the complicating variables which belongs to the set $Y = \{y | Ay = b, y \geq 0 \text{ and integer}\}$. The inner minimization is a continuous linear problem that can be

dualized by means of dual variables π associated with the constraint set $Dx = d - B\bar{y}$:

$$\max_{\pi \in \Re^{m_2}} \{\pi^T(d - B\bar{y}) : \pi^T D \leq c\} \tag{2.7}$$

Based on duality theory, the primal and dual formulations can be interchanged to extract the following equivalent formulation :

$$\min_{\bar{y} \in Y} \{f^T\bar{y} + \max_{\pi \in \Re^{m_2}} \{\pi^T(d - B\bar{y}) : \pi^T D \leq c\}\} \tag{2.8}$$

The feasible space of the inner maximization, i.e., $F = \{\pi | \pi^T D \leq c\}$, is independent of the choice of $\bar{y}$. Thus, if F is not empty, the inner problem can be either unbounded or feasible for any arbitrary choice of $\bar{y}$. In the former case, given the set of extreme rays Q of F, there is a direction of unboundedness $r_q, q \in Q$ for which $r_q^T(d - B\bar{y}) > 0$; this must be avoided because it indicates the infeasiblity of the $\bar{y}$ solution. We add a cut

$$r_q^T(d - B\bar{y}) \leq 0 \qquad \forall q \in Q \tag{2.9}$$

to the problem to restrict movement in this direction. In the latter case, the solution of the inner maximization is one of the extreme points $\pi_e, e \in E$, where E is the set of extreme points of F. If we add all the cuts of the form (2.9) to the outer minimization problem, the value of the inner problem will be one of its extreme points. Consequently, problem (2.8) can be reformulated as :

$$\min_{\bar{y} \in Y} \quad f^T\bar{y} + \max_{e \in E}\{\pi_e^T(d - B\bar{y})\} \tag{2.10}$$

$$\text{subject to} \quad r_q^T(d - B\bar{y}) \leq 0 \qquad \forall q \in Q \tag{2.11}$$

This problem can easily be linearized via a continuous variable $\eta \in \Re^1$ to give the following equivalent formulation to problem (2.1–2.5), which we refer to as the Benders *Master Problem* (MP) :

$$\min_{y,\eta} \quad f^T y + \eta \tag{2.12}$$

$$\text{subject to} \quad Ay = b \tag{2.13}$$

$$\eta \geq \pi_e^T(d - By) \qquad \forall e \in E \tag{2.14}$$

$$0 \geq r_q^T(d - By) \qquad \forall q \in Q \tag{2.15}$$

$$y \geq 0 \quad \text{and integer} \tag{2.16}$$

Constraints (2.14) and (2.15) are referred to as *optimality* and *feasibility* cuts, respectively. The complete enumeration of these cuts is generally not practical. Therefore, Benders (1962) proposed a relaxation of the feasibility and optimality cuts and an iterative approach. Thus, the BD algorithm repeatedly solves the MP, which includes only a subset of constraints (2.14) and (2.15), to obtain a trial value for the y variables, i.e., $\bar{y}$. It then solves subproblem (2.7) with $\bar{y}$. If the subproblem is feasible and bounded, a cut of type (2.14) is produced. If the subproblem is unbounded, a cut of type (2.15) is produced. If the cuts are violated by the current solution, they are inserted into the current MP and the process repeats.

Figure 2.2 illustrates the BD algorithm. After deriving the initial MP and subproblem, the algorithm alternates between them (starting with the MP) until an optimal solution is found. To confirm the convergence, the optimality gap can be calculated at each iteration. The objective function of the MP gives a valid lower bound on the optimal cost because it is a relaxation of the equivalent Benders reformulation. On the other hand, if solution $\bar{y}$ yields a feasible subproblem, then the sum of both $f^T\bar{y}$ and the objective value associated to the subproblem provides a valid upper bound for the original problem (2.1)-(2.5).

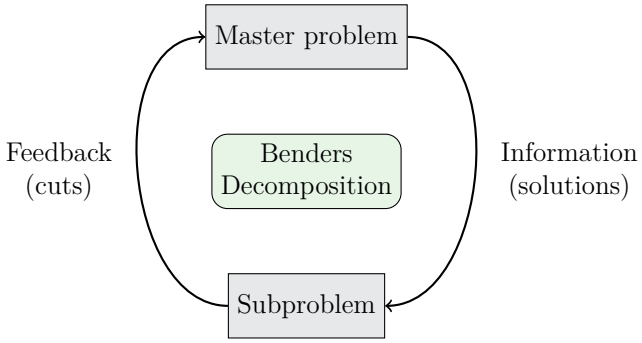

Figure 2.2 Schematic representation of the Benders decomposition method.

2.2.2 Model selection for Benders decomposition

A given problem can usually be modeled with different but equivalent formulations. However, from a computational point of view, the various formulations may not be equivalent. Geoffrion and Graves (1974) observed that the formulation has a direct impact on the performance of the BD. Magnanti and Wong (1981) demonstrated that a formulation with a stronger LP relaxation will have, in general, a better performance. This is because of the tighter root node and the smaller number of fractional variables and also because the generated cuts are provably stronger. Sahinidis and Grossmann (1991) proved that the BD method applied to

a mixed integer nonlinear programming (NLP) formulation with a zero NLP relaxation gap requires only the cut corresponding to the optimal solution to converge. Cordeau et al. (2006) studied a stochastic logistic network design problem. They found that when the original formulation was strengthened with a set of valid inequalities (VIs), the performance of the BD method was considerably improved.

These observations confirm the importance of tight formulations in the context of the BD method. However, tighter formulations are often obtained by adding additional (problem dependent) constraints. This may result in a more time-consuming subproblem, which may also exhibit a higher degree of degeneracy. Therefore, there must be a trade-off between the reduction in the number of iterations and the additional difficulty of the subproblem.

2.2.3 Relationship to other decomposition methods

The BD method is closely related to other decomposition methods for LP, such as Dantzig–Wolfe and Lagrangian optimization (see Lim (2010) for details). In particular, solving an LP by Dantzig–Wolfe decomposition is equivalent to applying the BD approach to its dual. The relationship is clear since Dantzig–Wolfe is particularly suitable for problems with complicating constraints, and the dual of these constraints will be the complicating variables in the BD method. Note that the subproblems are also equivalent in the two methods. The BD method is also equivalent to a cutting-plane method applied to the Lagrangian dual.

In integer programming, the situation is more complex, and there is no simple relationship among the decomposition methods. In contrast to Lagrangian relaxation and Dantzig–Wolfe decomposition, the BD method directly converges to an optimal solution to the MILP rather than to a relaxation of the problem ; therefore, there is no need to embed it in a branch-and-bound framework. However, the classical BD approach cannot handle integrality requirements in the subproblems ; variants have been proposed (Section 2.9).

Finally, there are close relationships between Benders cuts and various classical VIs (Magnanti et al., 1986). For instance, Costa et al. (2009) demonstrated that cutset inequalities are essentially Benders feasibility cuts, while Benders feasibility cuts are not, in general, metric inequalities and require additional lifting procedures for conversion into metric inequalities. Therefore, the classical BD method has several numerical and theoretical limitations, for which various enhancement and acceleration strategies have been proposed.

2.3 Taxonomy of the enhancement strategies

A straightforward application of the classical BD algorithm may require excessive computing time and memory (Magnanti and Wong, 1981; Naoum-Sawaya and Elhedhli, 2013). Its main drawbacks include : time-consuming iterations ; poor feasibility and optimality cuts ; ineffective initial iterations ; zigzagging behavior of the primal solutions and slow convergence at the end of the algorithm (i.e., a tailing-off effect) ; and upper bounds that remain unchanged in successive iterations because equivalent solutions exist.

Much research was dedicated to exploring ways to improve the convergence of the algorithm by reducing the number of iterations and the time required for each iteration. The former goal is aimed for by improving the quality of both the generated solutions and the cuts, and the latter by improving the solution procedure used to optimize the MP and subproblem in each iteration. The decomposition strategy that defines the initial MP and subproblem is another fundamental building block of the algorithm with significant consequences for its efficiency, as it determines both the difficulty of the problems and the quality of the solutions. We define therefore a four-dimension taxonomy, illustrated in Figure 2.3, that captures all these factors.

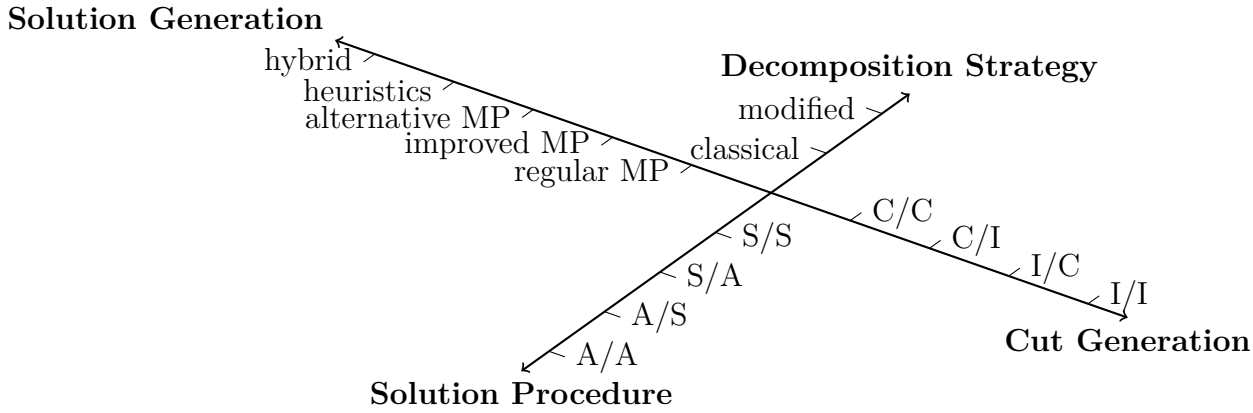

Figure 2.3 Components of taxonomy.

The *decomposition strategy* specifies how the problem is partitioned to obtain the initial MP and subproblem. In a *classical* decomposition all the linking constraints and noncomplicating variables are projected out. In a *modified* decomposition these constraints and variables are partially projected to maintain an approximation of the projected terms in the MP.

The *solution procedure* concerns the algorithms used for the MP and subproblem. The *standard* techniques are the simplex method and branch-and-bound. They are treated as black-

box solvers, and no attempt is made to adapt them to the characteristics of the problem or the convergence requirements of the BD algorithm. *Advanced* strategies exploit the structure of the MP and subproblem or the search requirements of the BD algorithm. For example, these strategies may aim to control the size of the problems or relax the requirement that they are solved to optimality at every iteration. We write "S/A" to indicate that standard techniques are used for the MP and advanced techniques are used for the subproblem. Similarly, we define A/A, A/S, and S/S.

The *solution generation* concerns the method used to set trial values for the complicating variables. The classical strategy is to solve the MP without modification (referred to as *regular MP*). *Heuristics*, an *alternative MP*, or an *improved MP* can be used to generate solutions more quickly or to find better solutions. *Hybrid* strategies can also be defined, e.g., one can use the regular MP to get an initial value for the master variables and then use heuristics to improve it.

The *cut generation* concerns the strategy used to generate optimality and feasibility cuts. Classically, this is done by solving the regular subproblem obtained from the decomposition. Other strategies reformulate the subproblem or solve auxiliary subproblems. The "C" and "I" symbols represent the *classical* and the *improved* strategies, respectively. For example, "C/I" indicates that the classical strategy is used to generate optimality cuts and the improved strategies are used to generate feasibility cuts. Sections 2.4 to 2.7 survey the strategies of each component of the taxonomy.

2.4 Decomposition strategies

Recent studies have presented various modified decomposition strategies. Crainic et al. (2016, 2014a) emphasized that the BD method causes the MP to lose all the information associated with the noncomplicating variables. This results in instability, erratic progression of the bounds, and a large number of iterations. Moreover, the problem structure associated with the linking constraints (2.3) is lost, and thus many classical VIs are not applicable. The authors proposed *Partial Benders Decomposition* strategies that add to the master explicit information from the scenario subproblems, by retaining or creating scenarios, or both. They obtained significant improvements in terms of number of generated cuts and computational time.

The *nonstandard decomposition* strategy of Gendron et al. (2016) is another interesting example of a modified decomposition. After decomposing the problem, the authors obtained a subproblem with integer variables and nonlinear constraints. To improve the convergence

of the algorithm, the authors retained the projected variables in the MP but relaxed the integrality requirement. They also included in the MP a linear approximation of the nonlinear constraints. They observed a significant improvement in performance, although the difficulty of the MP was noticeably increased.

As indicated by the results mentioned above and the limited studies conducted in this dimension (4.17% of the cited articles in this review paper), further research into decomposition strategies is worthwhile, as modified decomposition strategies may significantly strengthen the relaxed MP. Such approaches are not only complementary to adding VIs, discussed in Section 2.6.2, but may also provide the opportunity to derive a wider range of, possibly stronger, VIs.

2.5 Solution procedure

The iterative solution of the MP and subproblem is a major computational bottleneck. In particular, the MP, an MILP formulation, is often lacking special structure, and is continually growing in size becoming more and more difficult to solve. Classically, the MP is solved to optimality via branch-and-bound, while the subproblem is handled with the simplex method. In this section, we survey the various alternatives that have been proposed. These strategies exploit the structure of the MP and subproblem or are designed to improve the convergence speed. Figure 2.4 lists the strategies that we discuss.

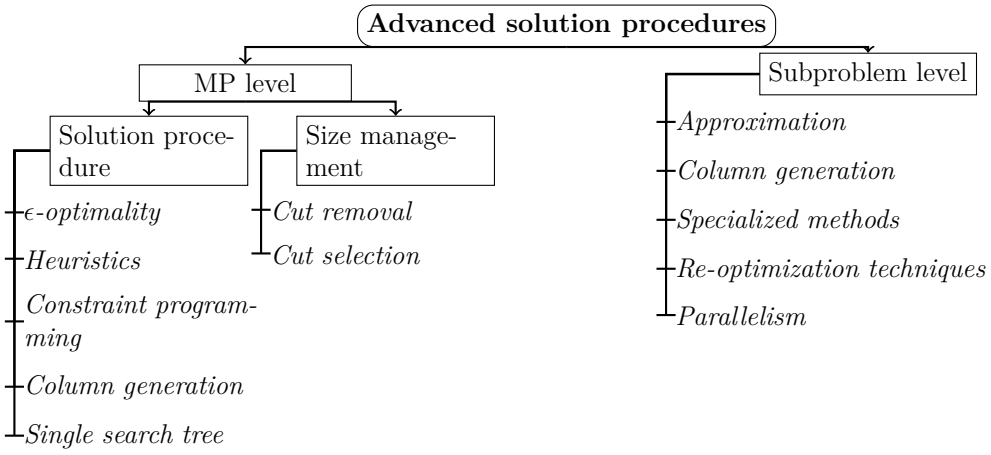

Figure 2.4 Advanced solution procedures.

2.5.1 MP level

It has often been reported that more than 90% of the total execution BD time is spent on solving the MP (Magnanti and Wong, 1981; Zarandi, 2010). The strategies proposed to partially alleviate this computational bottleneck, discussed in the next two subsections, either a) manage the size of the problem or b) solve it more efficiently.

Size management

In any optimal solution to the MP, the number of active constraints never exceeds the number of decision variables (Minoux, 1986). Thus, many of the generated cuts do not contribute to the convergence of the algorithm and merely slow it down (extra handling effort and memory limitations). Therefore, cut deletion or *clean-up strategies* are important, especially when multiple cuts are inserted into the MP at each iteration.

There is no reliable way to identify the useless cuts, so the clean-up strategies are heuristic (Ruszczyński, 2003). They usually inspect the slack values associated with each cut, a cut with a relatively high slack value over some predetermined number of iterations being a good candidate for removal. One can avoid the regeneration of eliminated cuts (and prevent the possible cycling of the algorithm) by keeping them in a pool and reinserting them into the MP whenever they are violated by the current solution. Cuts should be removed infrequently because constraint removal has a disturbing impact on off-the-shelf optimization solvers, especially when re-optimization techniques are used for faster solution of the MP ; see Geoffrion (1972) and Pacqueau et al. (2012) for implementation details.

Sometimes, multiple cuts are available for insertion but not all of them are worth adding. It is actually important to add cuts cautiously to avoid an explosion in the size of the MP. Holmberg (1990) defined *cut improvement* as follows : an optimality cut provides cut improvement if it is new and may be active in an optimal solution. Rei et al. (2009) used this definition when selecting solutions in order to generate cuts. They stated that a solution yields cut improvement if its value in the current MP is strictly smaller than the best known upper bound. Yang and Lee (2012) selected tighter (feasibility) constraints from a set of available cuts by measuring their distance from different interior points. After choosing a first cut from the pool, they added other cuts for which the "difference measure" from the first cut was greater than a user-defined threshold. This yielded considerable savings in computational time and number of iterations.

In summary, one can control the size of the MP by removing unnecessary cuts or avoiding the insertion of an available cut. However, these strategies are not often used. Considering

the articles cited in the present paper, only 3.13% of them used these strategies. One reason is that simultaneously inserting many cuts into the MP reduces the number of iterations, and this often compensates for the increase in the difficulty of addressing the problem. Another reason is that these techniques are heuristic, so they may remove necessary cuts or add cuts that prove unhelpful. There is a need for further research in this area.

Algorithm

It is not necessary to solve the MP to optimality at every iteration in order to achieve global convergence. Some researchers have focused on quickly finding suboptimal solutions. Others have taken advantage of the structure of the MP to solve it more efficiently.

Geoffrion and Graves (1974) were the first to address the MP's computational difficulties. They observed that it is not necessary to solve the MP to optimality at every iteration to produce valid cuts. In fact, there is no incentive to do so at the beginning of the algorithm because the relaxation is weak. They solved the MP to ϵ-*optimality* at each iteration, with ϵ decreasing as the algorithm proceeds to ensure global convergence. Lin and Üster (2014) terminated the algorithm whenever the MP could not produce feasible solutions in the presence of an ϵ-optimality constraint, indicating that the upper bound lies within ϵ% of optimality. Kewcharoenwong and Üster (2014) obtained an encouraging speedup with this approach in the context of fixed-charge relay network design in telecommunications.

An alternative is to solve the MP via *(meta-)heuristics*, which not only reduces the time but also allows the generation of multiple cuts per iteration, yielding faster improvement of the lower bound (Raidl, 2015). This strategy may lead, however, to worse bounds and a lack of control, which could prevent the generation of necessary cuts. For an MILP, the bounds may be worse than those of the LP relaxation of the original problem (Holmberg, 1994). Thus, the MP must be solved, at certain iterations, to optimality in order to ensure global convergence. However, fewer of these iterations are usually needed (Poojari and Beasley, 2009).

Constraint Programming (*CP*) is another possible approach. In a workforce scheduling application, Benoist et al. (2002) showed that CP can be a better choice than mixed integer programming (MIP) solvers, because of its greater ability to handle special constraints. Similarly, Corréa et al. (2007) considered the simultaneous scheduling and routing of automated guided vehicles, using CP to solve the scheduling MP and an MIP solver to handle the routing subproblem. They observed improvements of several orders of magnitude when the MP was solved by CP.

A few researchers have applied *Column Generation* (*CG*) to the MP to handle certain struc-

tures more effectively, aiming for tighter bounds at the root node of the branch-and-bound tree. Cordeau et al. (2001b) proposed using BD to handle linking constraints in a simultaneous aircraft routing and crew scheduling problem. They formulated an aircraft routing MP and a crew pairing subproblem. Because of their special structure, the linear relaxation of both problems were handled by CG (see Mercier et al., 2005, for a similar application). Restrepo et al. (2015) applied BD to solve a multi-tour activity scheduling problem ; the MP was handled by CG embedded in a branch-and-price framework because of its large number of variables. The integration of CG into the BD framework appears theoretically challenging because of the simultaneous addition of rows and columns that are often interdependent. The discussion of this issue is beyond the scope of this article ; see Muter et al. (2015) for further information.

In the classical BD, one MP (an MILP) is solved to optimality at each iteration. Each time, a new branch-and-bound tree is built and considerable time is likely spent revisiting candidate solutions that have been eliminated earlier. One can instead build a *single search tree* and generate valid cuts for the integer (and fractional) solutions encountered inside the tree, attaining the same optimal solution. This strategy, often referred to as *Branch-and-Benders-cut* (*B&BC*), yielded promising results (e.g., Fortz and Poss, 2009; Fischetti et al., 2010; de Camargo et al., 2011; Taşkın and Cevik, 2013; de Sá et al., 2013; Gendron et al., 2016; Crainic et al., 2014a; Pérez-Galarce et al., 2014). In addition to the numerical superiority of a modern implementation in comparison with the classical one, Naoum-Sawaya and Elhedhli (2013) showed that B&BC can make better use of the re-optimization tools of MILP solvers. Various strategies can be used to produce the cuts. For example, one can generate cuts in all feasible nodes or only when new incumbent solutions are found. It is necessary to establish a trade-off between cut generation and branching effort. For instance, Botton et al. (2013) studied when to generate cuts, their results indicating that generating cuts at every node of the search tree is inefficient because too many cuts are added and too much time is spent solving the subproblems. Better performance was achieved by finding as many violated cuts as possible at the root node and subsequently checking for violated Benders inequalities only at integer nodes.

We complete this section with two remarks. First, solving the MP via approximations, heuristics, or a single search tree may not be superior to the classical cutting-plane implementation, especially in applications where solving the MP takes significantly less time than the dual component of the algorithm. The modified approaches may then enumerate many solutions that are usually ignored by classical implementations. This is not the case, however, for most combinatorial optimization problems, for which the acceleration strategies we discussed are the most popular to handle the MP. Considering the cited researches throughout this article,

33.33% of them implemented one of these strategies. In particular, the single search tree has recently received considerable attention. This strategy leads to interesting research perspectives regarding the cut-generation strategies, the branching rules, the node selection, and the pruning strategies that have not been fully explored.

Second, CP has been shown to be better than MIP techniques for the MP if there are special constraints such as "all-different" constraints, logical relations, arithmetic expressions, integer division, and expressions that index an array of values by a decision variable. Finally, one can use alternative formulations for the MP to generate solutions more quickly, reducing the number of iterations. This topic is addressed in Section 2.6.1.

2.5.2 Subproblem level

The subproblem can be large, inherit complex features, or decompose into an excessive number of smaller subproblems. Various strategies have been proposed to solve the subproblem more effectively.

The solution of the subproblem may be extremely complex because it is a large-scale LP. Zakeri et al. (2000) showed that suboptimal solutions of the dual subproblem can be used to generate useful valid cuts. The authors observe that these inexact cuts are computationally less expensive and produce good results. In the same situation, commercial solvers (e.g., CPLEX) prefer the interior point approach to the simplex method. Thus, the BD method may converge to an incorrect solution, since the cuts are not necessarily associated with extreme points of the dual polyhedron (Yang and Lee, 2012). However, given the condition introduced by Zakeri et al. (2000) for inexact Benders cuts, convergence can still be guaranteed.

CG is another effective approach for large-scale linear subproblems with special structure (Cordeau et al., 2001b). Mercier et al. (2005) showed that for large subproblems complete enumeration is impossible, but that variables can be iteratively generated via CG. Similar applications of CG within a BD framework can be found in Cordeau et al. (2001a), Mercier and Soumis (2007), and Papadakos (2009).

Subproblems with special structure can often be solved efficiently. One can derive a closed-form solution or apply specialized algorithms. Fischetti et al. (2016) observed that the subproblem for the uncapacitated facility location problem can reduce to a knapsack problem, which has a closed-form solution. Similarly, Randazzo et al. (2001) obtained a series of trivial network flow subproblems with a closed-form solution. Contreras et al. (2011) obtained a semi-assignment problem for each commodity in their hub location application ; these problems could be solved more efficiently by a specialized method compared to an LP solver.

Kouvelis and Yu (1997) derived shortest-path subproblems that were solved with Dijkstra's algorithm.

The decomposition sometimes yields several independent subproblems. One may consider solving only a subset of the subproblems at each iteration, especially at the beginning of the algorithm. To the best of our knowledge, there is no such strategy for combinatorial optimization problems, but Fábián (2000) has studied a similar idea for convex programming problems. The algorithm initially calculates rough estimates of the function values and gradients. As it proceeds, the calculations become more accurate.

In some applications, many subproblems are similar. Considering that the algorithm updates only the objective function of the dual subproblem between two consecutive iterations, one can exploit these similarities using re-optimization techniques : given the solution to one subproblem, the next can be optimized in just a few iterations (see, e.g., Birge and Louveaux, 1997; Vladimirou, 1998). However, since the algorithm updates only the objective function of the dual subproblem between one iteration and the next, further investigation of re-optimization techniques is required.

When there are many subproblems, parallel computing techniques are often used : the subproblems are solved in parallel on different processors. One processor, the *master*, usually solves the MP and coordinates the other processors, the *slaves*, which solve the subproblems. The primal solution of the MP is passed to the slave processors, and the objective function and the dual information obtained from solving the subproblems is returned to the master processor. Experiments have shown that this strategy is effective (e.g., Ariyawansa and Hudson, 1991; Nielsen and Zenios, 1997; Linderoth and Wright, 2003; Wolf and Koberstein, 2013; Pacqueau et al., 2012). The literature discusses some of the algorithmic challenges of this approach. For example, Linderoth and Wright (2003) implemented asynchronous communications in which the MP is re-optimized as soon as $\lambda\%$ of the cuts are received. Dantzig et al. (1991) used dynamic work-allocation : the next idle processor gets the next subproblem based on a first-in-first-out strategy until all the subproblems are solved and the MP can be recomputed with the added cuts. Nielsen and Zenios (1997) exploited the structural similarities of the subproblems, applying an interior point algorithm on a fine-grained parallel machine. Vladimirou (1998) implemented a partial-cut aggregation approach to reduce the communication overheads. Chermakani (2015) observed that when the number of subproblems is considerably larger than the number of available processors, so that some subproblems must be solved sequentially, it may be better to aggregate some of the subproblems.

In summary, the simplex method is the most widely used algorithm for the subproblem. However, when the subproblem has special structure, specialized algorithms are a better

option. In either case, when the decomposition yields several independent subproblems, parallelization is the method of choice. There has been limited investigation of approximation and heuristic methods. Yet, when the subproblem is a large-scale LP that cannot be further decomposed and has no special structure, heuristics may yield considerable speed-up. Constraint programming has also been a prominent technique to solve subproblems with particular structures. This is discussed in Sections 2.9.1 and 2.9.2. In terms of statistics, 31.25% of the cited works in this article implemented one of the outlined acceleration strategies, among which parallelization and specialized algorithms are the most popular techniques.

2.6 Solution generation

The quality of the solutions for the set of complicating variables directly determines the number of iterations, as they are used to generate cuts and bounds. These solutions are traditionally found by exactly or approximately solving the regular MP. Three approaches have been proposed to improve the quality of the solutions or generate them more quickly : (1) using alternative formulations, (2) improving the MP formulation, and (3) using heuristics to independently generate solutions or to improve those already found. Figure 2.5 lists the techniques that we discuss. Note that, the strategies are not mutually exclusive, and hybrid approaches may work well. A complete presentation of the hybrid strategies is beyond the scope of this article, however, since the appropriate combination of strategies is problem-dependent.

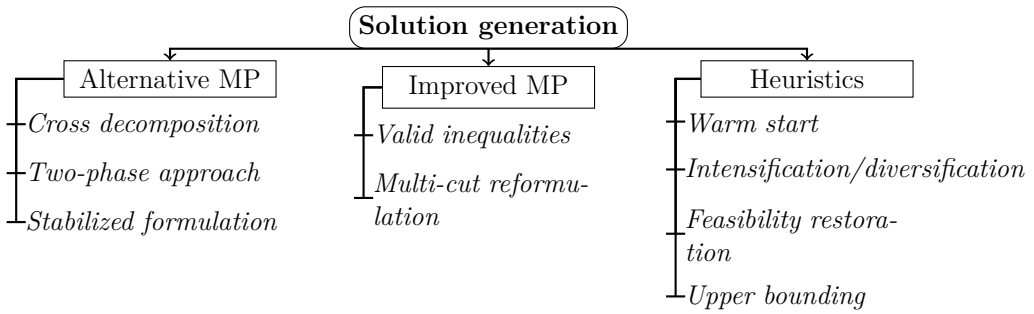

Figure 2.5 Strategies to generate solutions for the set of complicating variables.

2.6.1 Alternative formulation

Altering the MP, albeit temporarily, provides the means to address two main drawbacks in addressing it : (1) slow generation of solutions that may be poor-quality and (2) instability. The *two-phase* approach of McDaniel and Devine (1977) and the *cross decomposition*

of Van Roy (1983) address the first drawback. The former uses an approximation of the MP to generate solutions more quickly, while the latter uses an alternative MP to generate potentially better solutions.

McDaniel and Devine (1977) showed that valid Benders cuts can be obtained from the solutions to the LP relaxation of the MP. They applied the BD algorithm in two phases. In the first phase, the linear relaxation of the MILP MP is used to quickly generate solutions and tighten the relaxed MP, and in the second phase the integrality requirements are reintroduced and the solution process continues. The cross-decomposition algorithm (Van Roy, 1983) exploits the structure of both the primal and dual problems, combining the advantages of Lagrangian relaxation and BD. It alternates between two steps : (i) for a given y, the Benders subproblem (2.7) is solved to get the dual multipliers π, and (ii) for a given π, the Lagrangian subproblem is solved to produce a new y. After this alternation is terminated, the solutions of both the Benders and Lagrangian subproblems are used to generate new cuts for the Benders or Lagrangian MP. Again, convergence to optimality can only be ensured by periodically solving the Benders MP. However, doing this less often accelerates the solution process.

Instability is another widely recognized drawback of the regular MP (Birge and Louveaux, 1997; Zaourar and Malick, 2014), as it can yield excessively slow convergence. Two reasons are evoked for this phenomenon, the large initial steps and the excessive oscillation that occurs as the algorithm approaches the optimal solution (Rubiales et al., 2013). Several studies aimed to mitigate this undesirable behavior by adding constraints to the MP or changing the objective function. Regularized decomposition, trust-region, and level decomposition strategies have been used to obtain a stabilized MP formulation.

Regularized decomposition was introduced by Ruszczyński (1986) and extended by Ruszczyński and Świętanowski (1997). A quadratic term is added to the MP objective function to keep the solutions close to the current reference point, which is updated whenever certain conditions are satisfied. Computational results indicate that the method is efficient although it solves a quadratic problem rather than a linear one, since it decreases the number of expensive iterations (Ruszczyński and Świętanowski, 1997). The trust-region method can be thought of as a hypercube around a reference solution at iteration k, denoted y^k. The next solution must stay within this "trusted region." This is usually achieved by adding the constraint $||y - y^k||_\infty \leq \Delta$ to the MP ; the reference point y^k and the trust-region bound Δ will be updated during the algorithm (see Linderoth and Wright, 2003, for an implementation, computational results, and a proof of convergence). Level decomposition was developed by Lemaréchal et al. (1995) for nonsmooth convex optimization and adapted to stochastic

programming by Fábián and Szőke (2007). Its goal is to dampen the zigzagging behavior of the BD method with respect to the master solutions. This is achieved by solving a modified MP that minimizes the Euclidean distance of its solution y from the previous solution y^k (initialized to y^0), while ensuring that the approximate value of the next solution is not greater than a convex combination of the current best lower and upper bounds. A favorable comparison of this method with the standard BD method, regularized decomposition, and the trust-region approach was carried out by Zverovich et al. (2012).

The above stabilization techniques may not be directly applicable to combinatorial optimization contexts. Santoso et al. (2005) demonstrated that a trust region with the ℓ_2- or ℓ_∞-distance either yields a quadratic MILP or is not meaningful for a 0–1 MILP MP. Thus, they used a trust region that bounds the Hamming distance between the current MP solution and the previous one. However, since convergence cannot be ensured in the presence of this constraint, they used it only in the early iterations. Oliveira et al. (2014) observed an acceleration of up to 46% with the approach of Santoso et al. (2005). van Ackooij et al. (2015) instead stabilized the BD method via a proximal term (a trust region in the ℓ_1 norm) or a level constraint rather than adding quadratic terms to the objective function. In continuous problems the quadratic proximal term provides second-order information to the MP, but its impact in the combinatorial case tends to be insignificant. The authors showed the potential of stabilization techniques, particularly for large and difficult instances. Zaourar and Malick (2014) studied quadratic stabilization. This approach, common in the context of constraint decomposition methods, could drastically reduce the number of cuts, especially feasibility cuts. The computational time may not decrease, however, due to complexity of the method.

In summary, an alternative master formulation is used in more than 34% of the works cited in this article. The two-phase approach has become one of the most popular acceleration strategies for problems with time-consuming integer MPs (see e.g., Angulo et al., 2016; Rei et al., 2009; Papadakos, 2008; Cordeau et al., 2006; Costa, 2005; de Sá et al., 2013; Sen et al., 2015). Algorithms based on the single-search-tree strategy often use this approach to tighten the root node and reduce the size of the tree (see e.g., Botton et al., 2013). Cross-decomposition has received much less attention, although a recent study by Mitra et al. (2016) has demonstrated that it can be superior to the BD method when the underlying LP relaxation is weak. Stabilization techniques significantly reduce the number of iterations, but the cost of each iteration may increase because of the additional complexities that they introduce in the MP. This has usually prevented the use of stabilization techniques in combinatorial contexts.

Inexact methods offer a simple and efficient stabilization tool that may lead to a considerable time reduction. The success of these methods highlights the potential of heuristics, local

branching in particular, for solving the MP or more thoroughly exploring the neighborhood of the current solution to partially dampen the oscillations of the primal solutions.

Finally, as the instability of the classical BD method, especially in the early iterations, is mainly the result of a weak MP, strategies that strengthen the relaxation can mitigate the chaotic behavior. These strategies are the subject of the following subsection.

2.6.2 Improving the master formulation

After the projection and relaxation steps, the MP has no information on the subproblems and provides a weak approximation of the original feasible region. Therefore, we can expect erratic progression of the bounds and ineffective initial iterations. We can partially overcome these drawbacks by modifying the decomposition (see Section 2.4). The strategies discussed in this section will further improve the convergence rate (see e.g., Crainic et al., 2016, 2014a).

One way to strengthen the MP is to add VIs. Naoum-Sawaya and Elhedhli (2010) observed that this can significantly reduce the solution time and the number of generated cuts (feasibility cuts in particular). As a result, a wider range of instances can be solved. Saharidis et al. (2011) applied the BD method to a refinery system, adding two groups of VIs to the initial MP. The total time reduction ranged from 26% to 76%. Tang et al. (2013) used VIs to improve the initial lower bound by 21.39% to 208.95%. In addition, the number of instances solved increased by 30%. Adulyasak et al. (2015) studied a production routing problem and added lower-bound lifting inequalities to improve the initial lower bounds and solutions. These cuts provide information about the part of the original objective function that has been removed. The authors observed a significant reduction in the time, the optimality gap and the number of explored nodes. For other uses of VI see Taşkın and Cevik (2013), Kewcharoenwong and Üster (2014), Pishvaee et al. (2014), Jenabi et al. (2015), Emami et al. (2016), and Jeihoonian et al. (2016).

Feasibility cuts are undesirable because they do not improve the lower bound. In some applications, one can avoid unboundedness of the dual subproblem by including a set of VIs that exclude the infeasible solutions (see, e.g., Geoffrion and Graves, 1974; Birge and Louveaux, 1997; Contreras et al., 2011; de Sá et al., 2013). This avoids the burden of deriving the feasibility cuts. It can also reduce the cost of solving the MP, since the addition of both optimality and feasibility cuts makes it more difficult to solve (Wu et al., 2003).

Once we fix the complicating variables, it may be possible to make the decisions in the remaining problem independently. Thus, multiple smaller subproblems can be solved separately. The classic BD method inserts a single cut into the MP per iteration by aggregating the dual

information gathered from all the subproblems (Van Slyke and Wets, 1969). An alternative approach is to add a cut for each subproblem. This strategy, often referred to as *multi-cut reformulation*, generally outperforms the single-cut approach : it strengthens the MP more quickly and prevents the loss of information in the aggregation step (see, e.g., Contreras et al., 2011; Tang et al., 2013; Pishvaee et al., 2014; Sen et al., 2015; Jenabi et al., 2015). The size of the MP grows more rapidly, however, and the trade-off between the number of iterations and the computational time is problem-dependent. Birge and Louveaux (1997) gave the rule of thumb that the multi-cut is generally preferable when the number of subproblems is not much larger than the size of the dimension space of the master variables. Trukhanov et al. (2010) explored *partial cut aggregation*, in which the subproblems are divided into $|D|$ clusters and a cut is added for each cluster. They concluded that the best performance is attained for $1 < |D| < |S|$, where $|S|$ is the number of subproblems. They did not offer a specific strategy for the clustering. Brandes (2011) showed that clustering methods, in particular k-means and hierarchical clustering, can reduce the number of major iterations.

In summary, it is essential to strengthen the relaxed MP, and VIs are a powerful tool for this. Multi-cut reformulation is useful when the subproblem can be further decomposed into smaller problems. Using both strategies will often be more efficient. It should be noted that the reviewed strategies in this section have been used in 37.50% of the cited articles in this review paper. Finally, heuristics can be used to quickly tighten the MP by generating a set of initial cuts or multiple cuts per iteration. We discuss these strategies in the following subsection.

2.6.3 Heuristics

Numerous modifications may be needed to develop an efficient and competitive BD method (O'Kelly et al., 2014). Many researchers therefore apply heuristic procedures to generate solutions or, as a subordinate method, improve previously generated ones (Gelareh et al., 2015; Kewcharoenwong and Üster, 2014; Oliveira et al., 2014; Botton et al., 2013; Taşkın and Cevik, 2013).

Heuristics are widely used as a warm-start strategy to generate an initial set of tight cuts to strengthen the relaxed MP. Obviously, the selected heuristic should be appropriate for the problem at hand (Contreras et al., 2011). Lin and Üster (2014) observed that the initial selection of cuts is important in the context of wireless network design. They proposed a simple heuristic to generate feasible solutions and a set of good initial cuts. Easwaran and Üster (2009) used a tabu search as a warm-start meta-heuristic for a supply-chain network design problem ; the convergence rate and the size of the instances solved were considerably

increased. A different warm-start approach is to generate particular solutions. Randazzo et al. (2001) applied a shortest-path algorithm to obtain a special feasible solution to their local-access uncapacitated network design problem. Papadakos (2008) showed that convergence can be significantly improved when the algorithm starts from an initial point that lies in the interior of the MP solution domain rather than the initial MP solution itself.

Heuristics are also used to explore the neighborhood of the current MP solution as an intensification/diversification strategy. Rei et al. (2009) use a local branching heuristic to simultaneously improve the lower and upper bounds. They apply a limited number of local branching steps to either determine that the neighborhood contains no feasible solutions to the original problem or provide a pool of high-quality and diverse solutions. When the neighborhood is infeasible, they exclude the infeasible region by adding combinatorial cuts, which are a better alternative to classical feasibility cuts. When a pool of solutions is found, they are used to generate multiple optimality cuts. These cuts reduce the number of major iterations and cause the lower bound to increase more quickly. This strategy has also been applied to the closed-loop supply chain problem (Jeihoonian et al., 2016) and the sustainable supply chain network design problem (Pishvaee et al., 2014). Costa et al. (2012) gave general guidelines for the application of heuristics within the two-phase approach of McDaniel and Devine (1977). The goal is to quickly generate extra cuts associated with the heuristic feasible or infeasible solutions to reduce the need for integer iterations. The authors suggested using simple heuristics after each regular iteration of the BD method or whenever the incumbent solution is updated. The gains should compensate for the additional time spent on the heuristics.

Heuristics can be used to alleviate undesirable properties of the MP solutions. Wu et al. (2003) avoided the generation of feasibility cuts because they made the solution of the MP more expensive. It is possible in their application to acquire additional supply capacity at arbitrarily high prices, but convergence may be slow because of the side effects of big-M coefficients. The authors applied a heuristic to restore the feasibility of the solutions. The heuristic shifts excess demand from an infeasible subproblem to another source (i.e., to a different subproblem) so that a feasible solution can be found quickly. Emami et al. (2016) used a heuristic to extract feasible solutions from infeasible MP solutions, considerably improving convergence.

The upper bounds obtained from heuristics are often better than those obtained by the BD method itself, especially in the early stages of the algorithm. Roussel et al. (2004) successfully accelerated the BD method by applying tabu search to the original formulation to improve the upper bound. Santoso et al. (2005) stated that when the algorithm approaches an optimal solution, the various incumbent solutions differ in variables that have a small impact on the objective function, and so the upper bound changes little. They found that a simple fix-and-

optimize heuristic can yield a considerable acceleration. Improving the upper bound will also impact other parts of the algorithm. For instance, Contreras et al. (2011) observed that their heuristic not only improved overall convergence but also found better upper bounds that help with the reduction testing procedures. Pérez-Galarce et al. (2014) examined the interaction between the incumbent solution and the branching efforts. To improve the performance of their B&BC algorithm, the authors used a heuristic to enhance the incumbent solution.

In summary, heuristics are an important component of acceleration strategies. They are used in more than 25% of the cited articles in this review paper. Heuristics, even simple ones, can generate high-quality initial solutions and cuts, repair infeasible solutions, improve the quality of the MP solutions, reduce the computational cost of the MP and subproblem, and generate multiple cuts per iteration. Moreover, the BD method can be used to design effective heuristics ; see Section 2.8.

2.7 Cut generation

The number of iterations is closely related to the strength of the cuts, i.e., the values selected for the dual variables. Researchers have explored ways to select or strengthen the traditional feasibility and optimality cuts or to generate additional valid cuts.

Magnanti and Simpson (1978) and Magnanti and Wong (1981) were the first to consider the degeneracy of the subproblems, when the dual subproblem has multiple optimal solutions that do not yield cuts of equal strength. Hence, to find the strongest cuts, the solution of the dual subproblem must be judiciously chosen at each iteration. Magnanti and Wong (1981) selected a dual solution that dominates other possible cuts in terms of Pareto-optimality. A Pareto-optimal solution produces the maximum value at a core point $\hat{y}$, which is required to be in the relative interior of the convex hull of the subregion defined by the MP variables. After solving the regular dual subproblem (2.7), the authors solve an auxiliary subproblem of the form (2.17) to find the Pareto-optimal optimality cut.

$$\max_{\pi \in \Re^{m_2}} \{\pi^T(d - B\hat{y}) : \pi^T D \leq c, \quad \pi^T(d - B\bar{y}) = Q(\bar{y})\}, \tag{2.17}$$

where $Q(\bar{y})$ indicates the optimal cost of the regular subproblem for the current MP solution $\bar{y}$. Although this approach has proven effective in practice (e.g., Mercier et al., 2005), it must solve the secondary problem (2.17), which may be numerically instable and time-consuming. Additionally, it may be difficult to find a core point. This difficulty can be overcome by using approximate core points (Santoso et al., 2005), arbitrarily fixing components of the core-point vector (Mercier et al., 2005), or finding alternative points for a given problem

structure (Papadakos, 2008), although these methods do not guarantee the generation of Pareto-optimal cuts.

Papadakos (2008) proposed an algorithmic modification to circumvent the computational difficulties of Magnanti and Wong (1981)'s approach. The author showed that if a new core point is utilized at each iteration, Pareto-optimal cuts can be obtained from an independent formulation of the Magnanti–Wong cut generation procedure. This formulation removes the equality constraint in (2.17) that implies the dependency on the solution of the regular subproblem (2.7). The author showed that any convex combination of a valid initial core point and the MP solution gives an alternative Magnanti–Wong core point. de Sá et al. (2013) then showed that when the MP solution is rendered infeasible by the primal subproblem, a clever choice of the convex-combination weights can yield significant speed-ups. They chose the weights in such a way that the convex combination of the current MP solution with the previous Magnanti–Wong point results in a feasible subproblem.

Fortz and Poss (2009) and Naoum-Sawaya and Elhedhli (2013) generated Pareto-optimal cuts from the points obtained by an analytic-center cutting plane method. In other words, the extracted analytic centers are used as core points. Note that the dual subproblem is equivalent to that of Papadakos (2008), although this is not directly mentioned. The procedure has great potential for accelerating the classical BD algorithm in the context of capacitated facility location and multi-commodity capacitated fixed charge network design problems. However, its effectiveness depends on the quality of the core points and the efficiency of the re-optimization techniques. Moreover, similarly to the methodology of Magnanti and Wong (1981) and Papadakos (2008), they produce classical feasibility cuts based on the random selection of an extreme ray to cut-off the infeasibility.

Gelareh et al. (2015) generated analytic center points at every integer node of their B&BC algorithm to deal with degeneracy. They used analytic centers and their convex combination with the current MP solution to generate multiple cuts. The authors presented a box method to deal with degeneracy. A constraint is included in the auxiliary subproblem to bound the dual objective function, while the dual solution must be at least κ units distant from the actual optimal dual. The cuts produced are inexact, but the method performed well.

Sherali and Lunday (2013) developed the maximal nondominated cut generation scheme by formulating the cut selection as a multi-objective optimization problem. They showed that a small perturbation in the right-hand side of the primal subproblem is enough to give a maximal nondominated optimality cut, avoiding the need to solve the secondary subproblem as in Magnanti and Wong (1981) and Papadakos (2008). Given a goal-programming weight

$\mu > 0$, the dual subproblem is

$$\max_{\pi \in \Re^{m_2}} \{\pi^T(d - B\bar{y}) + \mu\pi^T(d - B\hat{y}) : \pi^T D \leq c\}, \tag{2.18}$$

Oliveira et al. (2014) considered the definition of the weight μ. The authors observed that the solutions obtained in the early iterations gave poor descriptions of the project cost, which is what the cuts attempt to approximate. They iteratively adjusted μ to favor solutions that focus on improving the original objective value $\pi^T(d - B\bar{y})$ rather than (2.18). To ensure convergence, the sequence $\{\mu^{(k)}\}_{k=1,\ldots,\infty}$ must satisfy $\sum_{k=1,\ldots,\infty} \mu^{(k)} \to \infty$ and $\mu^{(k)} \to 0$ as $k \to \infty$. The authors obtained results that compared favorably with those of Sherali and Lunday (2013) and Magnanti and Wong (1981).

Better feasibility cuts have also been investigated. Codato and Fischetti (2006) considered a binary problem where the BD method generates feasibility cuts exclusively. The authors observed that these cuts are weak because of the big-M constraints, and showed that stronger cuts, referred to as *combinatorial Benders cuts*, can be obtained by searching for minimal infeasible subsystems for the MP solutions. Experiments on two classes of mixed-integer problems indicated significant improvements in the bounds for the LP relaxation of the MP. Note that combinatorial cuts are not in general stronger than the classical feasibility cuts. Yang and Lee (2012) observed that the slow convergence of the BD algorithm is due to the selection of weak feasibility cuts. They extended the dominance rule of Magnanti and Wong (1981) in order to extract tighter feasibility cuts. However, this involves solving an auxiliary bilinear problem, which can be computationally expensive.

Fischetti et al. (2010) used an idea from Fukuda et al. (1997) : finding the most-violated optimality cut is equivalent to finding an optimal vertex of a polyhedron with unbounded rays. Fischetti et al. (2010) thus reformulated the subproblem as a feasibility problem where the optimality and feasibility cuts are derived by searching for minimal infeasible subsystems :

$$\max_{\pi \in \Re^{m_2}, \pi_0 \in \Re^1} \{\pi^T(d - B\bar{y}) - \pi_0\bar{\eta} : \pi^T D \leq \pi_0 c, \quad w^T\pi + w_0\pi_0 = 1\}, \tag{2.19}$$

where $\bar{\eta}$ is the current value of η, and w is a vector of normalization coefficients. The generated cut takes the form $\bar{\pi}^T(d - By) \leq \bar{\pi}_0\eta$. This approach simultaneously generates optimality and feasibility cuts without solving an auxiliary subproblem. It compared favorably with the classical cut selection scheme.

Some algorithms iteratively generate multiple cuts to obtain specific desirable characteristics. Saharidis et al. (2010) considered low-density cuts, i.e., cuts that include only a few MP variables. The ability of such cuts to strengthen the relaxed MP tends to be limited. To improve

these cuts, the authors developed a *covering cut bundle* cut-generation procedure. At each iteration, it produces a set of low-density BD cuts that cover $\alpha\%$ of the MP variables. The authors observed that adding several low-density cuts is better than adding a single high-density cut corresponding to the sum of the low-density cuts because it ensures a level of diversification in the cuts. Saharidis and Ierapetritou (2013) observed that it can be computationally less expensive to cover all the MP variables. Their strategy, referred to as *maximum density cut generation*, generates a cut that involves all the MP decision variables that are not covered in the BD cut. The authors observed that this significantly decreases the number of iterations and the time requirement for two different scheduling problems. Saharidis and Ierapetritou (2010) considered a case where obtaining optimality cuts using the classical BD method is hard. The bounds progress slowly because numerous feasibility cuts are generated before an initial feasible solution yielding an optimality cut is found. Whenever a feasibility cut is generated, they apply a *maximal feasible subsystem* to produce an optimality cut. This cut is produced by relaxing a minimum number of constraints in order to obtain a feasible subproblem. The authors observed significant improvements in convergence. The weakness of this strategy is that the reduction in the number of iterations may not always compensate for the additional time required to solve the auxiliary MILP subproblem.

In summary, the classical cut-generation scheme can be inefficient, particularly when the subproblems are degenerate or infeasible. Thus, 31.25% and 17.71% of the cited articles in this review paper have used one of the mentioned strategies to generate optimality and feasibility cuts, respectively. Almost every application of the BD that yields degenerate subproblems uses one of the strategies we have discussed to generate Pareto-optimal cuts. However, generating Pareto-optimal cuts may not yield a net computational advantage, since the reduction in the number of iterations might not compensate for the increase in the number of subproblems at each iteration (Mercier and Soumis, 2007). Strategies such as maximal nondominated cut generation may be more efficient since they eliminate the need to solve the auxiliary subproblem. Regarding the feasibility cuts, the strategies based on nondominated cuts have focused on optimality cuts ; feasibility cuts are found based on a random selection of the extreme rays. Only the strategy that generates combinatorial cuts for subproblems with big-M constraints has proven its worth in practice. Moreover, feasibility and optimality cuts are usually treated separately. To the best of our knowledge, only Fischetti et al. (2010) have developed a unified framework for both types of cuts. Clearly, further research is necessary.

2.8 Benders-type heuristics

Because of time and memory limitations, the execution of the BD method might be stopped before its convergence is established. it may not be possible to prove the convergence of the BD method. Moreover, in many practical applications, decision-makers do not need a provably optimal solution, a good feasible solution being deemed sufficient. Such a solution is often obtained somewhat early in the solution process.

From a heuristic point of view, the BD method is an attractive methodology because it can take advantage of special structures and provides a rich framework for the design of efficient search mechanisms (Côté and Laughton, 1984; Raidl, 2015). The method also overcomes many drawbacks of heuristics such as the inability to verify the solution quality and the difficulty to reduce the search space by using dual information (Easwaran and Üster, 2009; Boschetti and Maniezzo, 2009). These factors have promoted the development of algorithms that we refer to as *Benders-type heuristics*. We now discuss some of these.

Applying Lagrangian relaxation to the Benders cuts has been a popular approach, especially when the MP without cuts has a special structure (Minoux, 1984). Côté and Laughton (1984) applied Lagrangian relaxation to the feasibility and optimality cuts so that the remaining constraint sets have a special structure and specialized algorithms can be applied. Aardal and Larsson (1990) proposed a heuristic for a multi-item dynamic production planning problem. They created structured subproblems and MPs, priced out the BD cuts using Lagrangian multipliers in order to maintain the problem structure, and used a subgradient procedure to update the Lagrangian multipliers. The algorithm attained an average deviation of 2.34% from the optimum. Holmberg (1994) studied different approximations for the Benders MP, concluding that its Lagrangian dual cannot yield better bounds than the Lagrangian dual of the original problem even if all the feasibility and optimality cuts are present in the MP.

An alternative to the above approaches is to first apply Lagrangian relaxation and then use the BD method to optimize the Lagrangian dual subproblem. Pinheiro and Oliveira (2013) tackled problems with complicating constraints that are challenging for the BD method. They first applied Lagrangian relaxation to these constraints and then used the BD method to optimize the problem at each iteration of the dual Lagrangian algorithm. Wang et al. (2016a) applied a similar methodology to solve a corrective risk-based security-constrained optimal power flow problem. The results of both studies point to the ability of the approach to handle large-scale, complex problems. Further research in this area would thus be worthwhile.

One challenge in large-scale problems is the need to solve a sequence of difficult integer MPs. Many researchers have explored the use of meta-heuristics for the MPs. Poojari and Beasley

(2009) used a genetic algorithm combined with a feasibility pump. This enabled the authors to add multiple cuts per iteration, which yielded larger increases in the lower bounds. Although the MP was never solved to optimality, good results were obtained. Jiang et al. (2009) used a similar hybridization, based on tabu search, for multi-product distribution network design. A genetic-BD hybrid algorithm for vehicle routing and scheduling (Lai et al., 2012) and the capacitated plant location problem (Lai et al., 2010) has greatly reduced the computational time in comparison with the classical BD method. Boschetti and Maniezzo (2009) solved both the MP and the subproblem heuristically ; their algorithm was competitive with state-of-the-art meta-heuristics. Note that these strategies do not provide a valid lower bound, and thus, to assess the solution quality, the MP must be solved to optimality or approximated from below.

Proximity Benders is a decomposition heuristic proposed by Boland et al. (2015). The authors observed that the BD method rarely improves the incumbent solution, and finding good solutions may require considerable computing time. The authors used a *proximity heuristic* to more frequently improve the upper bound obtained from the sequence of MP solutions. Computational experiments demonstrated the potential of the method. Kudela and Popela (2015) proposed a genetic algorithm where the BD method is used to take advantage of the block structure. The authors reported favorable results in comparison with the genetic algorithm without the decomposition. Behnamian (2014) proposed a Benders-based variable neighborhood search algorithm for a multi-objective scheduling problem. The goal was to accelerate the assessment of the estimated improvement of each neighborhood. The new heuristic outperformed a variable neighborhood search, a tabu search, and a hybrid of these two methods, particularly on larger instances.

Another approach solves the LP relaxation of the MP and uses round-off heuristics to find an integer solution. Pacqueau et al. (2012) use the BD method to solve the linear relaxation and then fix some of the variables to their upper/lower bounds. Their algorithm iterates until an integer solution is obtained. They solved problems with up to 10 million integer variables in less than 27 minutes with an average accuracy of 0.2%, while CPLEX could handle only instances with fewer than 500000 integer variables. This highlights the potential of efficient Benders-type heuristics for problems with computationally intractable MPs, particularly those with tight linear relaxations.

In summary, the BD method enables heuristics to take advantage of special structures and use dual information. Many Benders-type heuristics either solve the MP heuristically or use approximate MP formulations that do not provide global convergence. Benders-type heuristics can handle a wider range of structures than the BD method. However, these algorithms

cannot find a provably optimal solution. We discuss extensions of the BD algorithm that can exactly solve a wider range of problems in Section 2.9.

2.9 Extensions of the classical Benders decomposition method

The classical BD algorithm was proposed for certain classes of MILPs for which the integer variables were considered to be complicating, and standard duality theory could be applied to the subproblem to develop cuts. Extensions of the method have allowed it to address a broader range of optimization problems, including integer subproblems (e.g., Carøe and Tind, 1998), nonlinear functions (e.g., Cai et al., 2001; Geoffrion, 1972), logical expressions (e.g., Eremin and Wallace, 2001), multi-stage programming (e.g., Lorenz and Wolf, 2015), and stochastic optimization (e.g., Van Slyke and Wets, 1969). When applied to stochastic problems the BD method is commonly referred to as *L-shaped decomposition*. It enables such problems to be decomposed by the realization of the uncertain parameters. Many algorithms for these important and challenging problems rely heavily on its premises (Ruszczyński, 2003). We have already discussed the literature on this variant, which is equivalent to the classical BD method. In this section, we discuss the extensions to problems with discrete subproblems, logical expressions, and nonlinear terms as well as multi-stage programming.

2.9.1 Discrete subproblems

When some of the projected variables are required to be integer, standard duality theory cannot be applied to derive the classical Benders cuts. A different theoretical framework or modifications to the generation scheme are needed to handle integer subproblems effectively.

When the complicating variables are required to take 0–1 values, one can use *lower-bounding functions* (LBF) instead of the regular optimality cuts (Laporte and Louveaux, 1993). These constraints enforce a change to the current solution or the acceptance of its associated cost. They usually take the form

$$\eta \geq (Q(\bar{y}) - L) \left(\sum_{a \in A_1} y_a - \sum_{a \in A_0} y_a - |A_1| \right) + Q(\bar{y}), \tag{2.20}$$

where $Q(\bar{y})$ is the cost of the subproblem for the given solution $\bar{y}$, A_1 and A_0 are respectively the variables with values of 1 and 0 in $\bar{y}$, and L is a lower bound on $Q(y)$ over y. The BD method with LBF cuts (2.20) is also applicable to problems where the subproblem can be evaluated with a closed-form analytical formula. Given the enumerative nature of (2.20), it is usually complemented with other VIs to improve the lower bound. A common strategy

is based on solving the linear relaxation of the subproblem to generate regular optimality cuts (Cordeau et al., 2001b; Mercier and Soumis, 2007; Papadakos, 2008). Moreover, the optimality cut (2.20) depends on the exact solution of $Q(\bar{y})$ and gives no useful information on the other y solutions. These issues are partly addressed by Angulo et al. (2016).

A similar variant of the classical BD method, often referred to as *combinatorial Benders decomposition*, likewise does not use the dual information to generate cuts. This variant can handle problems where the MP is a 0–1 integer program and the subproblem is a feasibility problem (i.e., a problem with no objective function). It excludes the current MP solution from further consideration via *combinatorial cuts*, which usually take the form

$$\sum_{a \in A: \bar{y}_a = 1} (1 - y_a) + \sum_{a \in A: \bar{y}_a = 0} y_a \geq 1, \tag{2.21}$$

Constraints of the form (2.21) are frequently used in the BD method as feasibility cuts. They are often strengthened according to the structure of the application, e.g., nonlinear power design in green wireless local networks (Gendron et al., 2016), lock scheduling (Verstichel et al., 2015), strip packing (Côté et al., 2014), and radiation therapy (Taşkın and Cevik, 2013).

Carøe and Tind (1998) used general duality theory to reformulate the subproblems, using VIs based on dual price functions to produce the BD cuts. They showed how such functions can be obtained when the subproblem is solved via standard techniques such as branch-and-bound or cutting planes. Sherali and Fraticelli (2002) considered applications with 0–1 mixed integer subproblems. They showed that the classical BD method is applicable if a convex hull representation of the constrained region is available. They employed the reformulation–linearization technique and lift-and-project cuts as a sequential convexification procedure for the subproblems. The cuts generated by these two methods were functions of the MP variables and were globally valid, which could lead to a finite convergent BD algorithm. Sen and Higle (2005) applied disjunctive programming to produce a convex characterization for the discrete subproblems. They showed that VIs generated for a given MP solution and a particular subproblem can be used to obtain VIs for any other solution or subproblem. This result can be used to define the cut-generation procedure in an overall BD approach. This approach was extended by Sen and Sherali (2006), who showed how branch-and-cut algorithms can be used for the subproblems.

We conclude with a remark on solving the integer subproblems. Heuristics have proven their worth in accelerating the BD method, particularly when the subproblem reduces to a feasibility-checking program or generates more feasibility cuts than optimality cuts (Os-

man and Baki, 2014). Heuristics can rapidly detect infeasibility and avoid the exact solution of difficult subproblems (e.g., Luong, 2015) or find approximate optimality cuts quickly (e.g., Raidl et al., 2014). In the latter case additional refinement is required, since the cuts may eliminate optimal solutions. On the other hand, CP is widely used to handle feasibility subproblems with special constraints because of its ability to handle those constraints and because it identifies infeasibility more quickly than traditional MIP-based approaches can (see e.g., Jain and Grossmann, 2001; Maravelias and Grossmann, 2004).

2.9.2 Logic-based Benders decomposition

There is a growing interest in optimization models that include logic relations. These models can usually be transformed into regular optimization models, but the extra variables and big-M constraints often yield a weak formulation. Furthermore, one cannot always obtain a continuous linear subproblem ; it may contain some integer variables and nonlinear functions. In these cases, standard linear duality cannot be used to develop classical BD cuts.

Hooker and Ottosson (2003) and Hooker (2011) introduced an extension known as *logic-based Benders decomposition* (*LBBD*). The LBBD method is similar to the classical BD method. It decomposes a given problem into an MP and one or many subproblems, and it uses constraint-generation techniques to gradually reduce the solution space of the relaxed MP. However, each subproblem is an "inference dual" problem that finds the tightest bound on the MP's cost function implied by its current solution. This bound is then used to generate cuts that are passed back to the MP. If the MP solution satisfies all the bounds produced by the subproblems, convergence has been achieved ; otherwise, the process continues.

A major advantage of the LBBD method is that the subproblem needs not take a specific form : it can be an MILP (Roshanaei et al., 2017), a CP (Hooker, 2005), an NLP (Wheatley et al., 2015), or a feasibility-checking problem (Harjunkoski and Grossmann, 2002). However, the LBBD method does not have a standard template for the production of valid cuts. Instead, they must be tailored to the problem at hand, typically based on knowledge of its structure. For some problems simple cuts exist (Hooker, 2007), but one must balance their effectiveness with the ease of extraction (Zarandi, 2010). It has been shown that The LBBD method can outperform state-of-the-art MIP and CP solvers in various applications, often by several orders of magnitude (e.g., Jain and Grossmann, 2001). It has been applied to a range of problems, including planning and scheduling (e.g., Hooker, 2007; Benoist et al., 2002), facility location/fleet management (Zarandi, 2010), radiation therapy (Luong, 2015), transportation network design (Peterson and Trick, 2009), and the minimal dispatching problem of automated guided vehicles (Corréa et al., 2007). It is worth mentioning that the B&BC

framework discussed in Section 2.5.1 is often referred to as the branch-and-check method in context of LBBD (Thorsteinsson, 2001; Beck, 2010).

2.9.3 Generalized Benders decomposition

Many optimization problems involve nonlinear functions and constraints. If the problem is easily linearized or the nonlinearity occurs only in the domain of the complicating variables, it can be solved via the classical BD method (Fontaine and Minner, 2014; Osman and Baki, 2014; Cai et al., 2001). Specifically, whenever the subproblem takes the form of a continuous linear formulation. Otherwise, an extended BD method is necessary.

Geoffrion (1972) proposed *Generalized Benders Decomposition* (*GBD*). It can solve nonlinear problems for which the subproblem is a convex program, because dual multipliers satisfying strong duality conditions can be calculated for such problems (Bazaraa et al., 2013). It is also particularly appealing for nonconvex nonlinear problems that can be convexified after fixing a subset of variables (Costa, 2005).

Sahinidis and Grossmann (1991) showed that the GBD method may not lead to a global or even local optimum for MINLP problems. Specifically, when the objective function and some of the constraints are nonconvex or when nonlinear equations are present, the subproblem may not have a unique local optimum and the MP may cut off the global optimum. Rigorous global optimization approaches can be used if the continuous terms have a special structure (e.g., bilinear, linear fractional, concave separable). The basic idea is to use convex envelopes (or underestimators) to formulate lower-bounding convex MINLPs. These are then integrated with global optimization techniques for continuous variables, which usually take the form of spatial branch-and-bound methods (see Grossmann, 2002, for further details). Similarly, Grothey et al. (1999) observed that a simplistic application of the GBD algorithm to a convex nonlinear problem may converge to a nonstationary point. They showed that the convergence failure results from the way in which the infeasible subproblems are handled, and they proposed a feasibility restoration procedure.

2.9.4 Nested Benders decomposition

The *Nested Benders Decomposition* (*NBD*) method is based on the idea of applying the BD method to a problem more than once. It is particularly appropriate for multi-stage (stochastic) problems (Birge, 1985) in which each pair of adjacent stages can be considered “separately". The NBD views the scenario tree as a set of nested two-stage problems and applies the BD method recursively. Every problem associated with an inner node in the tree

is both MP to its children and a subproblem of its parent. It is necessary to choose the sequencing protocols : after solving the problems at a given stage, one can either push primal information down toward the leaf nodes or pass dual information up toward the root node. This issue and some acceleration strategies are addressed by Wolf (2014).

The NBD method can also be applied to deterministic single-stage problems, particularly when one wishes to simplify the MP by reducing the number of integer variables. For example, Naoum-Sawaya and Elhedhli (2010) applied the BD method to obtain a binary MP and a mixed integer subproblem. They then applied the BD method to the subproblem to obtain an integer MP and a linear subproblem.

2.9.5 Bi-level programming and global optimization

Bi-level programming problems are often handled via the reformulation techniques that transfer the bi-level problem into single level using the Karush-Kuhan-Tucker (KKT) optimality conditions or strong duality from linear programming theory. This transformation, however, is not always possible, particularly when the second level is an MIP program. (Saharidis and Ierapetritou, 2009) employed the BD method to make the use of the KKT optimality conditions valid for resolution of MIP bi-level problems. In this method the decomposition creates a MIP master formulation and a bi-level continuous linear program which is amenable to the KKT optimality condition.

2.10 Conclusions

We have presented a state-of-the-art survey of the BD method. We have discussed the classical algorithm, the impact of the problem formulation on its convergence, and the relationship to other decomposition methods. We have developed a taxonomy to classify the literature on acceleration strategies, based on the main components of the algorithm, which provided rich guidelines to analyze various enhancements, identifying shortcomings, trends and potential research directions. We have also discussed the use of the BD to develop efficient (meta-)heuristics, described the limitations of the classical algorithm, and presented extensions enabling its application to a broader range of problems.

The BD method was originally proposed for MILPs with continuous subproblems, and it has since been extended to handle a wider range of problems such as nonlinear, integer, multi-stage, and constraint programming problems. Four main classes of acceleration strategies have been developed to enhance the classical algorithm : modifying the decomposition, solving the MP and subproblem more effectively, generating stronger cuts, and extracting better

solutions. The effectiveness of these strategies is problem-dependent, and a combination of them usually gives the best results. The BD method has also been used to develop efficient heuristics for complex problems, particularly those that numerically or structurally are out of reach of the method. This is not however to say that research into the BD is over. There are still many challenges and open questions.

Generally speaking, the BD method has been suitable for problems in which temporarily fixing the complicating variables makes the remaining problem significantly easier to handle by, e.g., becoming suitable for specialized algorithms or offering the opportunity to transform a nonconvex problem into a convex one. The BD method appeared particularly appropriate for problems with a "few" complicating (normally 0–1) variables and so many continuous variables that solving the problem as a whole is inefficient. There are many examples of such problems in stochastic programming. The range of problem settings addressed is also expanding, however. Moreover, many problems suffer from having weak linear relaxations and numerical instability as a result of big–M constraints and the binary variables used to turn them on and off. The BD method can handle such problems by moving the big–M constraints to the subproblems and using specialized cuts to represent them. The BD algorithm has also been applied to bilevel optimization problems that cannot be transformed via the Karush–Kuhn–Tucker optimality conditions into single-level problems (Saharidis and Ierapetritou, 2009). Moreover, there are interesting optimization problems for which some of the constraints are not known in advance and must be generated iteratively. In other cases the subproblem does not have an amenable formulation but can be obtained via a closed-form analytical formula. We are aware of only one survey article focusing on the applications of the BD method, and it restricted its scope to fixed-charged network design problems (Costa, 2005). There is certainly a need for a comprehensive synthesis of the various applications of the BD algorithm.

The acceleration strategies are all problem-dependent, so they are all part of the BD toolbox and their interconnections are important. A better understanding of these interconnections could have a considerable impact on the convergence rate. There is also a need for comprehensive research into the acceleration methodologies to better understand their limitations and implications. This is certainly true for more recent strategies, e.g., decomposition and cut generation schemes.

Strategies that tighten the MP usually add inequalities only once, before the initial iterations. Given the encouraging results obtained, it would be interesting to explore the use of more advanced cutting-plane methods to further tighten the MP at each iteration. The proper sequencing of the VIs and the classical BD cuts would be of great importance.

Tightening the subproblem is another effective acceleration strategy, since stronger cuts will be generated. We are aware of only one relevant study, Bodur et al. (2017) that iteratively generate Gomory mixed-integer cuts to tighten the subproblem. We encourage further research in this area. We note that one can solve the linear relaxation of the original problem with a cutting-plane method, adding VIs that involve the continuous variables. After the decomposition, these VIs will be moved to the subproblem, and this can yield a pronounced improvement in the quality of the Benders cuts. Moreover, one can use the partial decomposition to iteratively generate VIs for the subproblems, provided the subproblem retained in the MP gives VIs for the projected subproblems as well.

In terms of generating solutions for the set of complicating variables, we are not aware of any study showing how to obtain better cuts via a careful selection from the multiple optimal solutions of the MP or showing how to modify the MP to generate solutions with specific characteristics, e.g., geometrically centered solutions. These two ideas have been successfully applied in the context of Dantzig–Wolfe decomposition (e.g., Holloway, 1973; Nemhauser and Widhelm, 1971).

Sometimes the subproblem further divides into several independent subproblems that can be optimized concurrently. The literature on parallel algorithms for combinatorial optimization problems indicates that the parallel variants of the BD algorithm are still in their infancy. The current model is the master–slave paradigm, which is not the most efficient strategy (Crainic, 2015). Executing heuristics in the inner loop of the BD method or hybridizing the algorithm with other methods can yield tremendous improvements in the convergence rate. These approaches have been developed in a sequential framework, although they can be run almost independently of the main BD algorithm. Therefore, the development of new parallel algorithms, particularly in cooperative frameworks, would be worthwhile.

The BD decomposition often yields subproblems with identical dual polyhedra. In this situation, the solution of any subproblem gives valid cuts for the other subproblems. To the best of our knowledge, this information has not yet been used in an acceleration strategy. It should be interesting to develop a strategy in which only some of the subproblems are solved at each iteration and their solutions are used to generate cuts for other subproblems. Clearly, the main challenges are the selection of the set of representative subproblems and the demonstration of the convergence of the algorithm.

There has been limited research into stabilization techniques for the BD method in the context of combinatorial optimization. All the approaches surveyed were based on binary complicating variables, and state-of-the-art strategies for combinatorial problems, attempt to stabilize the MP only at the beginning of the algorithm. There is thus a need for more effective techniques

in more general settings. One can take advantage of stabilization strategies developed for continuous problems by first solving the linear relaxation of the MP. Although this idea has not yet been explored, it may prove effective.

There are various BD extensions for which researchers have explored ways to enhance them. We plan to survey these enhancements in a future article.

CHAPTER 3 BACKGROUNDS AND PRELIMINARIES

In this Chapter, we recall important definitions and concepts that we extensively use in the following Chapters to help the readers have a better insight on the methodologies that we develop in this dissertation. We first recall the mixed-integer programming tools in Section 3.1. In Section 3.2, we remind some important definitions and results from stochastic (integer) programming. In Section 3.3, we introduce the general concept of solution methods for SIPs. Next, we present an overview of the parallel computing techniques. Finally, we close this Chapter with a brief overview of network design problems as we extensively use them in assessing our algorithms.

3.1 Mixed-integer programming

Mixed-integer programming (MIP) is a form of optimization problems in which some or all of the variables are restricted to be integer. We assume that the MIP problem is linear, i.e., the objective function and the constraints are linear, which is known as *mixed-integer linear programming* (MILP). A canonical representation of an MILP is

$$z^* := \quad \min_{y,x} \; f^\top y + c^\top x \tag{3.1}$$

$$\text{s.t.} \quad By \geq b \tag{3.2}$$

$$Ty + Wx \geq h \tag{3.3}$$

$$y \in \mathbb{Z}^n_+, \; x \in \mathbb{R}^m_+ \tag{3.4}$$

with $f \in \mathbb{R}^n$, $B \in \mathbb{R}^{k\times n}$, $b \in \mathbb{R}^k$, $c \in \mathbb{R}^m$, $W \in \mathbb{R}^{l\times m}$, $h \in \mathbb{R}^l$, $T \in \mathbb{R}^{l\times n}$. *Mixed 0-1 linear programming* and *pure integer programming* are special cases of the above problem where the y variables are restricted to take binary values and $m = 0$, respectively. For ease of presentation, we only consider MILP in the present Chapter.

The MILPs, except for the trivial cases, belong to the NP-hard class of optimization problems. This means that generally no polynomial algorithm is known to solve MILPs. Moreover, the difficulty of solving them increases exponentially as the problem size grows (Nemhauser and Wolsey, 1988).

Many of the practical solution algorithms to address problem (3.1)-(3.4) rely on ignoring the integrality requirements of the y variables. This is commonly known as the *linear programming* (LP) relaxation (Jünger et al., 2009). The LP relaxation of (3.1)-(3.4) is significantly easier

to solve and more importantly it gives a lower bound on the optimal integer solution. The simplex method, due to Dantzig (1998), is the most common solution approach to solve the LP problems. To measure the quality of the lower bound obtained from the LP relaxation, an *optimality gap* is calculated as the relative distance of the LP optimal objective value from a known upper bound value.

Currently, the most successful (exact) method to address MILPs depends on iteratively solving the LP relaxation of the problem such that at each iteration a better approximation of the feasible region (3.3)-(3.4) is obtained (Conforti et al., 2010). This method is known as *branch-and-cut* approach, which is a combination of the *branch-and-bound* algorithm with the *cutting plane* method.

3.1.1 Branch-and-bound algorithm

Solving integer programs relies on the *branch-and-bound* (B&B) method due to Land and Doig (1960) and Little et al. (1963). It is based on an implicit enumeration of the integer solutions. The algorithm works with the LP relaxation of the problem. It creates a rooted *tree*, where the *root node* is the LP relaxation of the problem. It then explores the branches of this tree, where each *node* is a subproblem because it corresponds to a subset of the feasible region. After evaluating the subproblem at the current node, i.e., *bounding*, the node solution is checked against upper and lower estimated bounds on the optimal solution. Then, that node is discarded if it cannot produce a better solution than the best one found so far by the algorithm. This step is called *pruning*. If the node cannot be pruned, a *branching* step is performed by creating two (or more) new disjoint subsets of the feasible region (i.e., two nodes) by restricting the range of integer variables.

The algorithm starts by solving the LP relaxation of the problem at the root node and then it proceeds with branching. After evaluating any node, it will be pruned if it was infeasible, integer or its value function exceeds the upper bound. Otherwise, one (or several) index $a \in \{1, ..., n\}$ are selected such that $\bar{y}_a \notin \mathbb{Z}$ and two new nodes are created by adding two complementary constraints to the LP relaxation of the node. Various *branching rules* can be applied depending on the application. A standard format for creating the two new nodes in the branching step is to use following constraints $y_a \leq \lfloor \bar{y}_a \rfloor$ and $y_a \geq \lceil \bar{y}_a \rceil$.

Two key decisions regarding *variable selection* and *node selection* are to be made. The first addresses the selection of a fractional variable in the branching step. The latter addresses the order in which the nodes are explored. These two decisions define the *search strategy* by which the tree is explored. Various strategies can be considered for each decision. We refer the reader to Linderoth and Savelsbergh (1999) and Achterberg et al. (2005) for a comprehensive

treatment of such strategies.

If an integer node is found, its value provides an *upper bound* on the optimal solution. Also, the minimum value among all active nodes gives a *lower bound* on the optimal cost. Therefore, the algorithm stops when the optimality gap is within a desired threshold or there are no more subproblems to explore.

3.1.2 Cutting plane method

The *cutting plane* method for integer programming was first proposed in the seminal works of Dantzig et al. (1954) and Gomory (1958). Cutting plane is a technique to refine the feasible region of a mixed-integer program such that its LP relaxation provides a better (tighter) characterization of the convex hull of the true feasible set.

Cutting plane methods work with the LP relaxation of the problem. When this problem is solved and a non-integer value for a integer variable is obtained, an inequality, called *cut*, is extracted which violates the current fractional solution and satisfies all the (feasible) integer ones. This step is called *separation*. This process of solving the LP problem and adding cuts repeats until an integer feasible solution is found or it is proven that the problem is infeasible or no more cuts can be found.

In other words, recall that the convex hull of set Y is the set of all convex combination of its (extreme) points and it is denoted as $conv(Y)$, i.e., $conv(Y) = \{\sum_{i=1,\dots,|Y|} \lambda_i y_i | \sum_{i=1,\dots,|Y|} \lambda_i = 1 \text{ and } \lambda_i \geq 0 \ \forall i\}$. If Y is defined as a mixed-integer set, i.e., $Y = \{(y, x) \in \mathbb{Z}^n \times \mathbb{R}^m : Wx + Ty \geq h\}$, then its $conv(Y)$ is a polyhedron according to Meyer (1974). This means that we can write a mixed-integer program as an LP, i.e., $\min_{y,x}\{g(y, x) : (y, x) \in Y\} \equiv \min_{y,x}\{g(y, x) : (y, x) \in conv(Y)\}$, if description of its convex hull by means of a set of cuts is known.

The important questions in cutting plane methods are how to find a violating cut at each iteration and how to guarantee the finite convergence of the algorithm. From polyhedral theory, if $y \notin conv(Y)$, there always exists a hyperplane (i.e., a cut) that separates y from $conv(Y)$ (Nemhauser and Wolsey, 1988). Note that $conv(Y)$ is convex hull of set Y. To assure the finite convergence, the cut needs to be a facet of Y, because although there are infinitely many cuts, there are only a finite faces of a polyhedron.

It is generally very difficult to construct the convex hull representation of Y, because finding a class of facet-defining cuts and solving the separation problems, in general, can be as hard as solving the original problem. So, the primary goal of cutting plane methods is to find a good approximation of the convex hull.

Cuts can be *generic* or *problem specific*. The first class are based on algebraic arguments and does not depend on the problem structure. Thus, they can be applied to the LP relaxation of any integer (linear) program. The finite convergence of cutting plane method with generic cuts for some subclasses of integer problems is known, see for instance Gomory (1958), Balas et al. (1993) and Chen et al. (2011) for pure integer, mixed 0-1 integer problems, and general integer problems with bounded integer variables, respectively. The second class of cuts, also known as *valid inequalities*, exploit some structural properties of the problem and they are often stronger than the generic ones. Therefore, they are rarely used in general purpose solvers because they are valid only for some specific problems. However, they are widely used to tighten the LP relaxation, if available, see e.g., Chouman et al. (2017).

3.1.3 Branch-and-cut

Performance of the B&B method highly depends on the bounding step. Generally speaking, number of the explored nodes depends on the bound at each node, particularly the root node. Therefore, a cutting plane method is often used in conjunction with B&B in order to refine the representation of the feasible region at some nodes of the B&B tree so as to tighten the LP relaxation bound. To this end, a cutting plane method is used before branching occurs. This combination of the cutting plane method and B&B is known as *branch-and-cut* (B&C).

In other words, after solving the LP relaxation at a node of the B&B tree, cuts are (iteratively) added to the formulation in order to tighten its bound. Clearly, the bound improves and fractionality of the solution reduces. Therefore, it increases the chance to prune that node or it requires less subsequent branching steps. This strategy is common in almost every commercial solver and generally it is currently the most successful method to solve integer programs (Conforti et al., 2010).

The cuts generated at each node of the B&C tree can be *local* or *global*. If a cut is only valid to that node and its subsequent children, it is called local, while a global cut is valid to the original problem and thus all nodes of the B&B tree. Generally speaking, the global cuts are preferred over the local cuts due to their wider impact and entailing less complications in the backtracking steps. The decisions of where and how many cuts to generate are usually made empirically. Generally speaking, several iterations of the cutting plane method are executed at the root node, while fewer or no iterations are performed in deeper levels of the B&B tree. We note here that if cuts are only added at the root node of the tree before branching, the method is called *cut-and-branch*.

A variant of the B&C method is known as *delayed constraint generation* (Nemhauser and Wolsey, 1988). This method is particularly useful for problems where constraints need to be

generated on-the-fly because there are an exponential large number of them or they are not known beforehand. For example, some graph problems such as the traveling salesman problem involve the so-called subtour elimination constraints that grow exponentially with the size of the network. Only a small subset of these constraints are actually needed and adding all them to the formulation may even prevent solving the LP relaxation. In this framework, after evaluating each (integer) node and before branching, it checks if violated cuts are available to add to the formulation. To ensure the convergence to an optimal solution, all violated cuts need to be extracted.

3.2 Stochastic programming

In many applications, the actual value of data becomes known over time and the decisions that need to be made now depend on the value of these parameters. For instance, in a simple crop planning problem, the farmer has to make a strategic decision about how much land to devote to each crop prior to knowing the weather conditions. Stochastic programming offers a powerful tool to model such optimization problems (Kall et al., 1994; Birge and Louveaux, 1997). We note that there are other approaches for optimization under uncertainties such as probabilistic programming (Charnes and Cooper, 1959) and robust optimization (Ben-Tal et al., 2009). However, the description of these methods of our scope.

Uncertainty in the input data for an optimization model comes from unknown future events. For instance, in the crop planning example, the amount of rain in each session is unknown to the farmer when she makes her strategic decisions at the beginning of session. Unfortunately, these uncertainties are unavoidable because the farmer has no control over them. However, there usually exists some historical or simulated data about the uncertain parameters (e.g., amount of rain) that gives some statistical information to estimate the probability distributions for the uncertain data (Dupačová et al., 2000; Kaut and Wallace, 2003).

In stochastic programming, having the probability distribution is the main assumption. Given the probability distributions, the uncertain data can be taken into account by modeling the uncertain parameters as random variables (Shapiro et al., 2014).

To model stochastic problems, different objectives can be used. Among all, minimization of expected costs is the most common function ; see Sen and Higle (1999) for different objective terms. Such an objective is appropriate especially for problems that yield decisions to be used repetitively over a certain time horizon (Santoso et al., 2005). Therefore, the Law of Large Numbers justifies the expected value objective functions, because if the process repeats itself for many times, the average cost will converge to the true value. There are a vast number

of such situations in optimization problems; see, for example, Wallace and Ziemba (2005) and Uryasev and Pardalos (2013). *Two-stage stochastic programs* with recourse are the most widely applied and studied problems of this type (Shapiro et al., 2014).

In the *first-stage*, also called here-and-now, the decision maker must take a decision now, while not knowing the exact outcome of the uncertain parameters. In the *second-stage*, also known as recourse decisions, when the parameters have became known, the decision maker can take a recourse action to adjust her plan accordingly. For instance, in the crop planning problem, the allocation of land to different seeds must be made before knowing the amount of rain during the session, i.e., the first-stage. At the end of the session, the amount of rain is observed and the yields are known. Therefore, the farmer can take recourse actions to sell/store the additional grains or to buy from external sources to cover the deficits for the next session.

The two-stage stochastic programming provides a set of implementable strategies specifying how to react when a particular random value is observed, which, appropriately emulates the decision-makers behavior. Therefore, the goal is to obtain a solution that, on average, performs well in the long-run. To do so, all possible realizations of the random variables need to be considered. This, however, entails a noticeable computational challenge because it requires to calculate the expected value function. This computational challenge depends on the dimension of the random variables. However, even for small dimensions, the exact calculation of the objective function may numerically be very intense. Therefore, the uncertain parameters are characterized with a finite set of discrete realizations, called *scenarios*. As an example see Crainic et al. (2011).

A common approach to sample the scenarios is Monte Carlo sampling. The resulting problem for the chosen set of the scenarios is usually known as *sample average approximation* (Rubinstein and Shapiro, 1990). In this method the set of scenarios are sampled such that they, collectively, closely approximate the (continuous) distribution. Each scenario is associated with a positive weight to reflect its occurrence probability. Once the appropriate set of scenarios is generated, the stochastic problems can be modeled as a deterministic optimization problem. We should note that in this method the optimization problem is solved with a fixed set of scenarios. In *stochastic approximation* method, however, the set of scenarios is iteratively sampled during the optimization procedure (Nemirovski et al., 2009).

The quality of the sample average approximation highly depends on the number of scenarios created to estimated the uncertain parameters. Clearly, larger number of scenarios give a better approximation of the original stochastic program, but solving the associated deterministic optimization problem would become very challenging due to its size. Moreover, to measure

the quality of the solutions obtained from the approximated problem, statistical lower and upper bounds on the optimality gap are needed. This requires solving multiple deterministic problems associated with different scenario samples. Therefore, it is very important to have efficient algorithms to get good solutions and to be able to evaluate the candidate solutions in a reasonable amount of time.

In many applications of stochastic programming, some discrete decisions have to be made. For instance, in a simple portfolio management problem, the investor has to decide whether or not to buy a specific asset prior to knowing the return rate for each asset. This combination of stochastic programming and integer programming is known as *stochastic integer programming* (Louveaux and Schultz, 2003; Ahmed, 2010). This class of optimization problems is very challenging to solve, because they are generally NP-hard due to their combinatorial nature and also they have a large-scale size due to the data uncertainty. Thus, it is no surprise that effective solution algorithms are scarce.

3.2.1 Two-stage stochastic integer programming

In this dissertation, we mainly focus on the two-stage stochastic integer programming. Thus, in this part, we recall some of the main backgrounds that we use in this dissertation. We refer the reader to the introduction by Ahmed (2010), the surveys by Louveaux and Schultz (2003) and Sen and Higle (2005), the books by Birge and Louveaux (1997) and Kall et al. (1994), for a comprehensive treatment of this subject.

In the two-stage stochastic programming, versus the multi-stage case, the uncertainty is observed only once and the decisions are made before and after observing the uncertainties. Let $\boldsymbol{\xi}$ be vector of random variables governed by probability function P with support Ω which is a closed set in the real space. We consider a canonical representation of these optimization problems with the following form, called *extensive formulation*

$$z^* := \min_{y} \; f^\top y + \mathbb{E}_{\boldsymbol{\xi}}[Q(y,\xi)] \tag{3.5}$$

$$\text{s.t.} \quad By \geq b \tag{3.6}$$

$$y \in \mathcal{Y} \tag{3.7}$$

where for each realization ξ of the uncertainty set

$$Q(y,\xi) = \min_{x} \; c_\xi^\top x \tag{3.8}$$

$$\text{s.t.} \quad W_\xi x \geq h_\xi - T_\xi y \qquad (\alpha) \tag{3.9}$$

$$x \in \mathcal{X} \tag{3.10}$$

with $f \in \mathbb{R}^n$, $B \in \mathbb{R}^{k\times n}$, $b \in \mathbb{R}^k$, $c_\xi \in \mathbb{R}^m$, $W_\xi \in \mathbb{R}^{l\times m}$, $h_\xi \in \mathbb{R}^l$, $T_\xi \in \mathbb{R}^{l\times n}$. Here, $\mathcal{X} \subseteq \mathbb{R}^m$ and $\mathcal{Y} \subseteq \mathbb{R}^n$ are nonempty closed subsets which define the nature of the x and y decision variables in terms of sign, bounds and integrality restrictions. In this program, y represents the first-stage decisions and x represents the second-stage decisions. We seek a feasible solution that minimizes the first-stage cost $f^\top y$ plus the expected cost of the second-stage decisions.

Note that the decisions are made in sequence. The second-stage problem takes the first-stage decisions y as fixed and determines the best recourse action associated with that decision for each realization of the uncertainties. Among all these possible solutions, the one that entails minimum first-stage and expected second-stage cost is chosen. Therefore, it provides a set of implementable strategies specifying how to react when a particular scenario is realized, which, appropriately emulates the decision-maker's behavior.

A common assumption in stochastic programming is to assume that the recourse problem is *bounded* from below. Thus, for any first-stage solution $\bar{y}$ and realization of the uncertainty ξ, we have $Q(\bar{y},\xi) > -\infty$. Another common assumption is known as the relatively *complete recourse property* which means that the recourse problem is always feasible for every first-stage solution. An extended version of this property is *complete recourse property* which means that the recourse problems are feasible for any $y \in \mathbb{R}^n$. These properties usually exist in many applications. However, if the problem does not have such properties, it can easily be achieved by adding nonnegative dummy variables to the recourse constraints (3.9) and penalizing them in the objective function (3.8). Another common assumption in stochastic programming is to assume that the recourse matrix W_ξ in the second-stage problem is independent of the uncertainty, i.e., $W_\xi = W$ for every ξ. Such problems are called stochastic programs with *fixed recourse*. These properties are usually important in the development of some algorithms, because they allow a convenient characterization of the feasibility region in computational steps.

The solution of program (3.5)-(3.7) is often referred to as *here-and-now* because we take the strategic decisions before realization of the uncertainty. On the contrary, if we wait until the uncertainty unfolds and then take the strategic decisions, i.e., first-stage decisions, such solution is referred to as *wait-and-see*. The latter provides a lower estimation of the optimal

cost of the former, which is very weak if the uncertainty has noticeable impact on the decisions (Wets, 2002).

To generate the appropriate set of scenarios, the common practice is to approximate the probability distribution P of ξ by a discrete probability distribution with finite support. This gives a finite set of scenarios each representing a possible realization of the random event. Given the scenario set S, we denote the probability of occurrence for scenario s by $p_s := P(\xi^s) > 0$, and the corresponding data by $(c^s, W^s, T^s, h^s) := (c_\xi, W_\xi, T_\xi, h_\xi)$ for every s in S such that $\sum_{s\in S} p_s = 1$. Accordingly, the extensive formulation, also called *deterministic equivalent formulation*, can be written as follows

$$EF := \quad \min_{y} \; f^\top y + \sum_{s\in S} p_s {c^s}^\top x^s \tag{3.11}$$

$$\text{s.t.} \quad By \geq b \tag{3.12}$$

$$y \in \mathcal{Y} \tag{3.13}$$

$$T^s y + W^s x^s \geq h^s, \qquad \forall s \in S \tag{3.14}$$

$$x^s \in \mathcal{X} \qquad \forall s \in S \tag{3.15}$$

where x^s defines the recourse variables for scenario $s \in S$. The objective function (3.11) minimizes the total cost consisted of first-stage cost and the expected second-stage cost. Constraints (3.12) and (3.13) enforce the first-stage requirements and constraints (3.14) and (3.15) make sure that the second-stage constraints are satisfied for every scenario.

3.3 Decomposition methods

The deterministic equivalent formulation (3.11)-(3.15) is a standard mixed-integer optimization program. Therefore, one can attempt to directly solve it. For example, if the problem is a linear continuous program, i.e., $\mathcal{Y} = \mathbb{R}^n$ and $\mathcal{X} = \mathbb{R}^m$, it might be solvable with a simplex or an interior point method. If it is a linear integer program, it can be solved with a branch-and-bound type algorithms or MIP solvers such as Gurobi or CPLEX. However, these methods generally fail due to the large-scale of the formulation. Although heuristics may seem the natural methodology of choice (Crainic et al., 2011), they are also subject to failure because of the same reason and more importantly, they provide no proof of convergence.

In this regard, decomposition techniques are essential in solving stochastic programs. Each decomposition method is based on four main mechanisms : (i) how the problem is divided into smaller subproblems, (ii) what information is generated and shared among the subproblems, (iii) how the results are combined to build a complete solution for the original problem

and (iv) how the search continues. For some decomposition examples, see Laporte (1992), Ruszczyński (1997), Lahrichi et al. (2015), Qi et al. (2015) and Chaieb et al. (2015).

The stochastic program (3.11)-(3.15) has a so-called *dual decomposition structure* and the dual of its LP relaxation has a *block angular structure*. In this dissertation, we are interested in mathematical decomposition methods which exploit these primal or dual structure of the problem, yielding the Benders decomposition (BD) algorithm or Lagrangian dual decomposition (LDD) method.

3.3.1 Benders decomposition algorithm

In this section, we consider two-stage stochastic integer programs with continuous recourse. We specifically consider the formulation (3.11)-(3.15) with $\mathcal{X} = \mathbb{R}^m_+$. We shall discuss variants of the BD algorithm which relax this requirement, see Chapters 2 and 5. Various solution algorithms have been designed to solve this problem, see, for example, Wets (1983), Kall (1994), Birge and Louveaux (1997), Carøe and Schultz (1999) and Ruszczyński (2003). Most of these solution procedures rely heavily on the premises of the Benders decomposition method (Benders, 1962). Note that the application of BD to stochastic problems is often referred to as the *L-shaped method* due to Van Slyke and Wets (1969).

The BD method is to tackle problems with *complicating variables*, which, when temporarily fixed, yields a problem significantly easier to handle. It is based on a pattern of projection, outer approximation and relaxation (Geoffrion, 1970a,b). First, the model is projected onto the subspace defined by the first-stage variables. The projected term is then dualized and an equivalent model is built by enumerating all extreme points and rays of the dual polyhedron. The extreme rays indicate feasibility requirements for the first-stage variables (i.e., *feasibility cuts*) and the extreme points state the projected cost (i.e., *optimality cuts*). Enumerating all the extreme points and rays of the dual polyhedron is not computationally practical and most of the associated cuts will not be active at an optimal solution. Therefore, a relaxation of the equivalent formulation is performed such that it initially includes no cuts, these being iteratively generated by sequentially solving a *Master Problem* (MP) and one or several *Subproblems* (SPs).

For a tentative value of the first-stage variables $\bar{y}$, the recourse problem (3.8)-(3.10) is a continuous linear program. Given a dual variable α associated with constraint (3.8), the dual

formulation of $Q(\bar{y}, s)$ is,

$$(SP(\bar{y}, s)) \quad Q(\bar{y}, s) = \max_{\alpha} (h^s - T^s \bar{y})^\top \alpha \tag{3.16}$$

$$\text{s.t.} \quad W^{s\top} \alpha \leq c^s \tag{3.17}$$

$$\alpha \in \mathbb{R}^l_+. \tag{3.18}$$

The above program is either unbounded or feasible and bounded. In the former case, the $\bar{y}$ solution is infeasible and thus there exits a direction of unboundedness $r_{q,s}$, $q \in F^s$ that satisfies $(h^s - T^s y)^\top r_{q,s} > 0$, where F^s is the set of solutions of the cone defined by $W^{s\top} \alpha \leq 0$ and $\alpha \in \mathbb{R}^l_+$, i.e., the set of extreme rays of the polyhedron defined by (3.17) and (3.18). To assure the feasibility of the y solutions, we need to forbid the directions of unboundedness through imposing $(h^s - T^s y)^\top r_{q,s} \leq 0,\ q \in F^s$ on the y variables. In the latter case, the optimal solution of the subproblem is an extreme point $\alpha_{e,s},\ e \in E^s$ of the polyhedron defined by $W^{s\top} \alpha \leq c^s$ and $\alpha \in \mathbb{R}^l_+$, where E^s is the set of extreme points of this polyhedron. We can thus rewrite the extensive formulation (3.5)-(3.7) by interchanging $Q(y, s)$ with its dual formulation, i.e., $SP(y, s)$, and ensuring the feasibility of the y solutions

$$MP := \min_{y} f^\top y + \sum_{s \in S} p_s \max_{e \in E^s} \{(h^s - T^s y)^\top \alpha_{e,s}\} \tag{3.19}$$

$$\text{s.t.} \quad By \geq b \tag{3.20}$$

$$(h^s - T^s y)^\top r_{q,s} \leq 0 \qquad s \in S, q \in F^s \tag{3.21}$$

$$y \in \mathcal{Y} \tag{3.22}$$

If we capture the value of inner maximization in a single variable θ^s for every $s \in S$, we can obtain following equivalent reformulation of the extensive problem (3.5)-(3.7), called *Benders equivalent reformulation*

$$MP = \min_{y} f^\top y + \sum_{s \in S} p_s \theta^s \tag{3.23}$$

$$\text{s.t.} \quad By \geq b \tag{3.24}$$

$$(h^s - T^s y)^\top r_{q,s} \leq 0 \qquad s \in S, q \in F^s \tag{3.25}$$

$$(h^s - T^s y)^\top \alpha_{e,s} \leq \theta^s \qquad s \in S, e \in E^s \tag{3.26}$$

$$y \in \mathcal{Y}. \tag{3.27}$$

Enumerating all the extreme points E^s and extreme rays F^s for each $s \in S$ is computationally burdensome and unnecessary. Thus, Benders (1962) suggested a relaxation on the *feasibility*

cuts (3.25) and *optimality cuts* (3.26) and a delayed constraint generation method. The method starts from MP formulation with initially no optimality and feasibility cuts, i.e., $E^s = \emptyset$ and $F^s = \emptyset$ for every $s \in S$. At iteration t, the obtained solution y^t from solving MP is fixed in subproblem $SP(y^t, s)$, $s \in S$ to extract an extreme point/ray and thus add the corresponding optimality/feasibility cut to the MP formulation. This alternation between the MP and the subproblems continues until the lower and upper bounds collide. To calculate these bounds, notice that the objective value of the MP at each iteration and $f^\top y^t + \sum_{s\in S} p_s SP(y^t, s)$ give the lower and upper bounds on the optimal value.

The aforementioned BD method is known as *multi-cut* reformulation as it adds a cut to the MP per scenario SP. However, if we aggregated the cuts into a single constraint, it is commonly referred to as *single-cut* version.

The BD method has been the subject of extensive study to improve its numerical performance, extend its application to a wider range of optimization problems and even to design new solution algorithms. Due to the extensive use of this method in this dissertation, we provide a comprehensive literature review of this method in Chapter 2.

3.3.2 Lagrangian dual decomposition algorithm

Carøe and Schultz (1999) developed an algorithm based on a specific reformulation of the extensive formulation and Lagrangian relaxation. We refer to this method as *Lagrangian dual decomposition* (LDD) method. It breaks the problem into $|S|$ disjoint subproblems where each subproblem is a single-scenario form of the extensive formulation. This is achieved by duplicating the first-stage variables, i.e., $y^1, y^2, ..., y^{|S|}$. According to the *nonanticipativity* of the first-stage decisions, these variables need to take the same value, i.e., $y^1 = ... = y^{|S|}$. Relaxing constraints enforcing this requirement by Lagrangian duality yields the respective decomposition method.

In other words, we rewrite our two-stage stochastic program of interest as follows :

$$\min_{y, y^1, ..., y^{|S|}} \quad \sum_{s\in S} p_s [f^\top y^s + {c^s}^\top x^s] \tag{3.28}$$

$$\text{s.t.} \quad By^s \geq b \qquad s \in S \tag{3.29}$$

$$y^s \in \mathcal{Y} \qquad s \in S \tag{3.30}$$

$$W^s x + T^s y^s \geq h^s \qquad s \in S \tag{3.31}$$

$$x^s \in \mathcal{X}^s \qquad s \in S \tag{3.32}$$

$$y = y^s \qquad s \in S \qquad (\pi^s) \tag{3.33}$$

This formulation is referred to as the *scenario formulation* or *split-variable formulation*. Relaxing the nonanticipativity constraint (3.33) using Lagrangian multipliers $\pi^s \in \mathbb{R}^n$ for each $s \in S$, we get following Lagrangian dual subproblem for each $s \in S$

$$L(\pi^s) := \min_{y^s \in Y^s, y \in \mathbb{R}^n_+} \{(p_s f - \pi^s)^\top y^s + p_s Q(y^s, s) + {\pi^s}^\top y\} \tag{3.34}$$

where $Y^s = \{y^s | By^s \geq b, y^s \in \mathcal{Y}\}$ for every $s \in S$. Note that $L(\pi^s)$ is a piecewise concave function of π^s and the last term can easily be solved by inspecting the coefficients. The Lagrangian dual problem is thus to find the multipliers that maximize the sum of $L(\pi^s)$ functions over S, i.e,

$$LDD := \max_{\pi^s, ..., \pi^{|S|} \in \mathbb{R}^n} \sum_{s \in S} \min_{y^s \in Y^s, y \in \mathbb{R}^n_+} \{(p_s f - \pi^s)^\top y^s + p_s Q(y^s, s) + {\pi^s}^\top y\} \tag{3.35}$$

We observe that for a given $\pi^1, \pi^2, ..., \pi^{|S|}$ the minimization problem can be solved disjointly for each $s \in S$. To solve the Lagrangian dual problem, Carøe and Schultz (1999) employed a subgradient method. In this case, a duality gap may arise since the nonanticipativity constraints are not necessarily satisfied for a mixed-integer problem. Thus, a branch-and-bound method is used to close the gap. Note that solving linear continuous problems by Lagrangian dual program using a cutting plane method is equivalent to solving its dual with the BD method, see Lim (2010) for complete details.

The LDD method is frequently referred to as *scenario decomposition* in the context of stochastic programming (Ahmed, 2013). This method is one of the most popular approaches in stochastic programming not only because it breaks the extensive formulation into more manageable pieces that can be solved in parallel but also due to the creation of subproblems that structurally resemble the original extensive formulation (Ruszczyński, 1997; Rockafellar and Wets, 1991). In addition, it offers the ability to address problems with integer recourse variables (Zou et al., 2016). This method entails interesting theoretical results which have motivated the development of both exact and heuristic methods ; see Fisher (2004), Rush and Collins (2012) and Koko (2013) for a more detailed information.

In practice, the LDD method usually reaches a lower bound which is much tighter than the LP relaxation bound. This bound is equal to solving the LP relaxation of the extensive formulation over $\bigcap_{s=1,2,...,|S|} conv(X^s)$ due to Theorem 6.2 in Nemhauser and Wolsey (1988), where $X^s := \{(y^s, x^s) | By^s \geq b,\ W^s x^s + T^s y^s \geq h^s,\ y^s \in \mathbb{Z}^n_+,\ x^s \in \mathbb{R}^m_+\}$. However, this requires updating the Lagrangian multipliers by subgradient (Held et al., 1974) or cutting planes (Mitra et al., 2016) which can be a bottleneck of the algorithm due to the slow convergence. Acceleration techniques have thus been developed for which examples include

the bundle method (Crainic et al., 2001), the volume algorithm (Barahona and Anbil, 2000), and analytical center cutting plane method (Goffin et al., 1992).

At each iteration, the LDD method yields a pool of integer solutions obtained from solving the Lagrangian dual subproblems (3.34). These solutions can be used to derive near-to-optimal upper bounds (Ahmed, 2013). In some cases, however, it might be hard to derive a good feasible solution from these solutions because they may not satisfy the nonanticipativity constraints (3.33). Crainic et al. (2011) used a heuristic scheme to update the first-stage costs iteratively in order to attain a consensus for the first-stage values among all the scenario subproblems. In a later study, Crainic et al. (2014b) argues that superior performance can be reached if a subproblem is associated with a cluster of scenarios rather a single scenario.

The LDD method cannot offer a provable convergence for non-convex problems, although in practice it might find very close-to-optimal ones. Convergence for mixed-integer problems might be difficult to obtain and it is even more unlikely when integer variables are indicator variables (i.e. their value has a great impact on other variables). Strategies have thus been designed to overcome this issue. Schütz et al. (2009) solved the resulting Lagrangian dual program by a cutting plane method in a bundle method with box constraints. Ahmed (2013) proposed a LDD-based convergent method using valid inequalities to prevent revisiting generated integer solutions at each iteration. Note that this algorithm can work independently of updating the Lagrangian multipliers. Branch-and-fix (Alonso-Ayuso et al., 2003) is another convergent LDD-based method which removes the nonanticipativity constraints and gradually enforces them inside the search tree. Branch-and-price algorithms can also be used to solve the Lagrangian dual problem and converge to a global optima (Lulli and Sen, 2004).

3.4 Parallel computing

Time requirement to solve many combinatorial optimization problems can disappointingly be large. Especially, in stochastic (integer) programming, one needs to iteratively solve many subproblems, usually, for a large number of iterations. *Parallel computing* can offer huge advantages to cope with this problem by, for example, solving the disjoint subproblems concurrently.

Parallelism has been employed for many years, but interests in it have grown lately due to the developments in computers and increasingly need from science and engineering communities. In this part, we attempt to provide the definition of parallel computing, how it is achieved, what are parallelization strategies, and how to measure the performance of parallel algorithms. We refer the reader to the excellent works by Gendron and Crainic (1994),

Crainic and Gendreau (2002), Ralphs et al. (2003), Talbi (2006) and Crainic (2017) for a comprehensive literature review of this topic with extensive examples and details.

3.4.1 Source of parallelism

Parallel computing is based on dividing the problem into smaller disjoint ones which can be solved in parallel. This follows from a decomposition of the total computational workload and distribution of the resulting tasks among available processors. Given a problem instance and an algorithm to solve it, the decomposition of total computational load may concern the problem's domain or the solution algorithm itself. In other words, if the computational workload is decomposable, it is usually done in one of the following ways :

— the solution domain is divided among processors, known as *data parallelism* or *search-space decomposition*,

— the algorithm tasks themselves are divided among processors, known as *functional decomposition*.

Domain decomposition may concern the problem's data or problem's structure. The former refers to the case where the problem-instance data is divided among several processes by assigning each process with the data on which it operates. A process may require data from several other processes thus, *communications* are required to respond to the requests. A classical example of data parallelism can be observed in solving finite difference equations. In this problem, there is a mesh of points and at iteration t of algorithm, following equation needs to be calculated :

$$x_{i,j}^{t} = \frac{x_{i-1,j}^{t-1} + x_{i,j-1}^{t-1} + x_{i+1,j}^{t-1} + x_{i,j+1}^{t-1}}{4} \tag{3.36}$$

As depicted in Figure 3.1, the domain is divided among three processors where each processor runs the same process but on different data. Therefore, the mesh points on the boundary of region assigned to a processor must share data in order to perform an iteration.

In the second case, the decomposition is generally implemented by partitioning the vector of decision variables (or attributes). For instance, in rich VRP problems, one can partition the solution space by only considering a set of particular attributes (e.g., client-day or client-depot) in each subproblem while the variables associated with the other attributes are fixed to a certain value. Or, it can be achieved by utilizing projection or relaxation techniques presented in the previous section.

In the functional decomposition, the focus is on the computations to be performed rather than manipulating data, i.e., it seeks computing-intensive and disjoint tasks in the inner loop

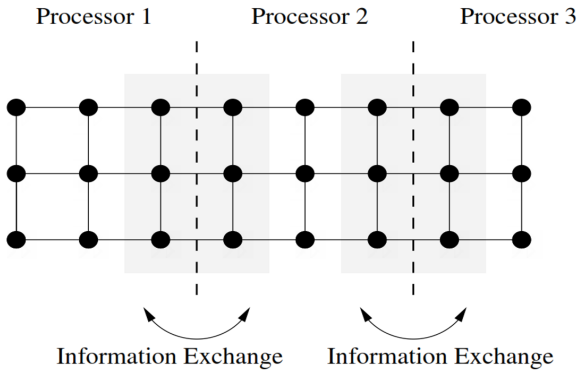

Figure 3.1 Illustrative example of data-parallelism and communication (Linderoth, 1998)

of algorithm to be performed in parallel by several processors. For example, in the presented decomposition methods, each processor receives a subset of the subproblems associated with the current solution, solve them and, returns the results.

Another source of parallelization is based on non-decomposition *multi-walk* strategies, which does not properly fit into these two lines of parallelization. Because no explicit attempt to decompose the total computational workload is carried out. This strategy mainly aims at providing a higher search robustness, particularly when various search parameters and randomness are involved in the algorithm. For example, several independent algorithms, each with different search mechanism, can be launched to address the same problem instance rather than distributing (or in some sense reducing) the total computational load. These algorithms may or may not communicate. Figure 3.2 gives a general picture on sources of parallelism.

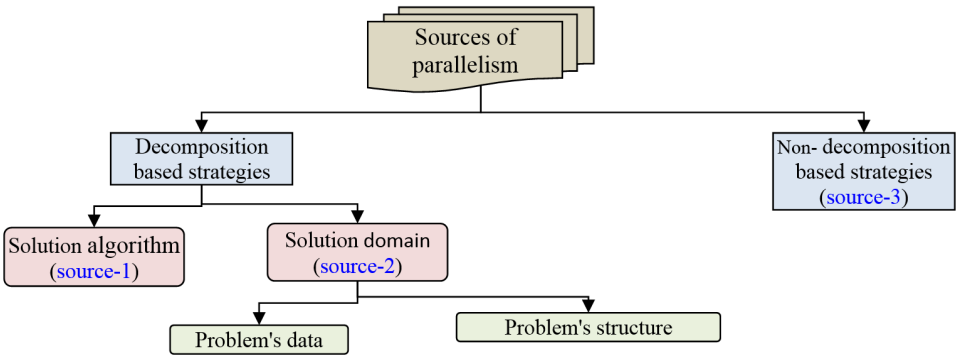

Figure 3.2 Main sources of parallelism for combinatorial optimization problems

Using these sources of parallelism, one can design a wide variety of parallel solution algorithms. In the rest of this section, we attempt to describe such algorithms. We first present

a taxonomy of the parallelization strategies that clearly illustrates the main components of these algorithms.

3.4.2 Taxonomy and paradigms

We follow the three-dimension classification of parallel algorithms presented by Crainic and Toulouse (1998). This taxonomy is based on the number of processors that *control the search*, the *communications strategies* used, and *variety of algorithms* running in parallel.

The first dimension in designing any parallel algorithm is "how the global problem solving is controlled ?". This refers to the so-called *search control cardinality*, indicating whether the global search is controlled by a single processor, *1-control* (1C), or by several processors that may collaborate or not, *p-control* (pC). Note that considering parallel B&B algorithms, these two parameters are frequently referred to as centralized or distributed (also called collegial) control.

The second dimension addresses "how information is exchanged ?". In other words, every parallelization necessitates some kinds of communications to provide the necessary data for computation or control the global search. Therefore, the second dimension is called *search control and communications* addressing the issue of information exchange. In this context, one generally refers to *synchronous* and *asynchronous* communications. In the former case, all concerned processes stop and engage in some form of communication to exchange information at the pre-specified moments determined either by hard-coded or by a control (master) process. While in the latter case, communications may occur at any time and depend only on the local algorithmic logic of the interacting processes. On the other hand, there is a possibility to work on the shared information to derive additional knowledge. Therefore, one can further classify this dimension based on the quantity and quality of the exchanged information. These levels are defined as follows. Note, in order to make the presentation clear, we provide a brief discussion on the possible combination of this parameter with the search cardinality parameter.

- *Rigid synchronization*, RS : the information is exchanged in its original form and at the predefined moments. In 1C/RS there is a central master process that sends the tasks to the slaves, and waits until all slaves finish their task before performing next iteration. On the other hand, in pC/RS strategy several searches are separately conducted with no communication among the various processors except at synchronization point.
- *Knowledge synchronization*, KS : a predefined number of explorations has been performed and possibly shared information is modified or used to infer new knowledge. The difference between rigid and knowledge synchronization in a 1C control context

is based on how much work the master assigns to each slave, while for p-C strategies it corresponds to the absence or presence of inter-process communications and knowledge exchanges.

— *Collegial*, C : communications between processes are established without prior agreement (asynchronously). In 1C/C strategy that follows master-slave model, each slave will be assigned with a work unit as soon it becomes idle. In pC/C strategy several search threads run independently and asynchronously communicate to share information.

— *Knowledge collegial*, KC : asynchronous communications with creation of new information from exchanged data. The only difference from the previous category is that in this category new knowledge based on the exchanged information is inferred.

In line with this subject, the way information is accessed is important. In this context, there are two possible options : *shared memory* or *distributed memories*. In shared memory there is a common "central memory", referred as blackboard, where every thread can write/read information on/from it. In the latter strategy, every process (or possibly a set of processes together) has its own memory and they communicate to each other accordingly to a pre-specified communication topology (e.g. a complete-graph connection scheme or a ring topology).

The third and last dimension to classify parallel algorithms is referred as *search differentiation*, because more than one solution method or variant (e.g. same algorithm with different parameter setting or different initial point) may be involved in the search. It specifies if search threads start from the same/different point, use same/different search strategy, and how many different algorithms are parallelized. Figure 3.3 represents this three-dimension taxonomy of parallelization strategies.

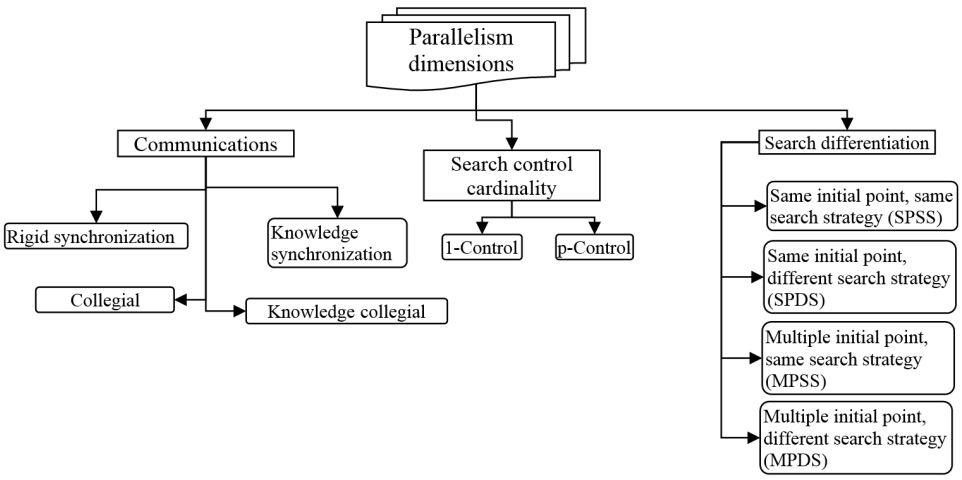

Figure 3.3 Main sources of parallelism for combinatorial optimization problems

Combination of the various sources of parallelism with different parallelization strategies gives a broad range of possibilities to design parallel algorithms. However, these algorithms can be accounted for with a relatively small number of frameworks (paradigms) :

— *Low-level parallelism* : they aim at accelerating the resolution process without modifying the algorithmic logic or the search space. They seek computing-intensive and disjoint tasks in the inner loop of algorithm to be performed in parallel by several processors, e.g., solving Benders subproblems on various processors. Moreover, they are implemented following the classical master-slave parallel programming model.

— *Multi search* : this strategy consists in running several search algorithms simultaneously and independently, each exploring the entire solution domain of the problem but with a different search trajectory.

— *Cooperative search* : this paradigm goes one step further than the previous strategy and integrates the mechanism to share the information obtained from the diversified search algorithms while the exploration is in progress. Sharing and eventually creation of new information yields a cumulative value with better output than the independent multi-search parallelism.

— *Integrative search* : in this class of parallelism the domain or search-space is divided into smaller sub-regions and then, each of the resulted subproblems is solved by a sequential algorithm, the (partial) solutions are collected to build a complete one and the decomposition is modified (if required).

— *Hybrid parallelism* : any combination of the previous paradigms. This strategy is also often called *hierarchical* parallelism.

3.4.3 Efficiency measurement

The traditional goal of designing parallel algorithms is to reduce the time required to solve a given problem instance. For the algorithms that run until the optimal solution is found, this translates into the well-known *speedup* measure. Absolute speedup with $\mathcal{P}$ processors is defined as ratio of the time required by best serial algorithm over that obtained by parallel algorithm using $\mathcal{P}$ processors. For the sake of simplicity, *relative speedup* is often used and means the ratio of the time taken for solving an instance on one processor over that required to solve the same instance using $\mathcal{P}$ processors. Speedup measures are more difficult to define when the parallel and sequential algorithm does not yield the same solution quality. In this circumstance, time comparison is not fair. Thus, the parallel methods should outperform their sequential counterparts in terms of solution quality and, ideally, computational efficiency, or, the additional time requirement has to be justified by higher quality solutions.

Efficient utilization of additional resources in parallelism is important. Thus, efficiency is computed as the speedup divided by the number of processors. When efficiency is equal/greater to one it is normally referred to as *linear/superliner speedup*. However, if this ratio is inferior to one, it exhibits the speedup anomalies due to *overheads*. More precisely, parallel algorithms might create redundant and/or unnecessary tasks. On the other hand, the algorithmic design might cause delays and idle times. These yield speedup anomalies and when efficiency is smaller than one, *parallel overhead* is measured as "total run time×(1-efficiency)".

3.5 Application problem

In this dissertation, *network design* (ND) constitutes one of our main application in numerically assessing the algorithmic developments, because of its vast applicability, its generality (i.e., appearing in the backbone of many optimization problems), and its complexity. We thus aim at reviewing some of the main aspects of the ND problems. However, the literature is so broad that a complete treatment of the topic is a significant endeavour by itself and clearly well beyond the scope of this document. Our central goal is to briefly discuss the general concept behind ND problems, illustrate their practical importance, and highlight some of the main challenges in solving them. We refer the reader to Magnanti and Wong (1984); Crainic and Laporte (1997); Gendron et al. (1999); Crainic (2000); Minoux (2001); Wieberneit (2008); Klibi et al. (2010) for a more elaborated and comprehensive treatment of this subject.

3.5.1 Network design problems

The ND problems are central to a large number of real-world applications including transportation, telecommunication, logistics, manufacturing, power systems, just to name a few (Klibi et al., 2010). Generally speaking, these problems revolve around the selection of a subset of (capacitated) arcs out of a potential set in order to support the flow requirements between various origin-destination (OD) pairs. The decisions are to be made such that an effective trade-off between the cost of design decisions and their impact on the performance of the resulting network is achieved, while demand requirements and a set of problem-specific constraints are satisfied.

The ND problems are defined on a graph, $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, consisting a set of nodes $\mathcal{N}$ (representing cities, terminals, depots, power plants, etc.) and a set of links $\mathcal{A}$ (representing roads, cables, pipelines, flight legs, services, etc.), where each link connects a specific pair of nodes. Links can be unidirectional (arc) or bidirectional (edge) and defined with various characteristics, e.g., capacity, cost, length, etc. The underlying goal is to se-

lect (build/install/acquire/upgrade) a subset of these links so as to move some entities (e.g., electricity, goods, people, messages, data) between some properly chosen pairs of nodes (e.g., supplier and vendor) in order to meet a set of specific requirements (e.g., following prescribed traffic requirements, achieving a level of security, and so on). These actions need to be done as efficiently as possible (e.g., minimum costs or travel times, maximum coverage or profit, and so on). Probably, the simplest example of the ND problems is the shortest spanning tree problem, which consists of determining a minimal length tree linking all nodes in a given graph.

There are various common and basic characteristics in almost every ND formulation. Broadly speaking, every formulation for these problems includes information on each of the three following important *parameters* : (1) parameters specifying the intensity level of communications to be established between pair of nodes (e.g. demand), (2) parameters specifying the amount of available resources (e.g. capacities), and (3) parameters indicating the costs of creating and using the network (e.g. fixed and flow costs). In terms of *decision* variables, network design problems usually involve two groups of decisions : *design* and *flow*. The former defines the structure and characteristics of the network, while the latter specifies how the network is used to satisfy the specified goals and requirements (Crainic and Laporte, 1997; Minoux, 2001). The main goal is to optimize both decisions simultaneously while satisfying a set of constraints. Hence, the fundamental trade-offs are usually between the fixed-charge costs and the variable (flow) costs.

3.5.2 The multicommodity capacitated fixed-charge network design problem

The *multicommodity capacitated fixed-charge network design* (MCFND) is a generic ND problem that properly captures many of the fundamental characteristics of the ND problems (Gendron, 2011). In the MCFND problem, a set of commodities $\mathcal{K}$ are to be simultaneously routed on a directed graph consisting a set of nodes $\mathcal{N}$ and a set of potential arcs $\mathcal{A}$. Each commodity $k \in \mathcal{K}$ has a demand of $d^k \geq 0$, which is to be routed from a unique origin node $O(k) \in \mathcal{N}$ to a unique destination node $D(k) \in \mathcal{N}$. The goal is to select a subset of the arcs to meet all demand at minimum cost, computed as the sum of the fixed design costs, charged whenever an arc is used, and the transportation costs. The unit transportation cost on arc $a \in \mathcal{A}$ for commodity k is denoted c_a^k, and the fixed design cost for arc $a \in \mathcal{A}$ is represented by f_a. Also, there is a capacity limit u_a on each arc $a \in \mathcal{A}$, limiting the maximum flow traversing it.

To model the MCFND as an MILP, we define binary design variables y_a indicating if arc $a \in \mathcal{A}$ is used (1) or not (0) and a flow variable $x_a^k \geq 0$ to reflect the amount of flow on arc

$a \in \mathcal{A}$ for commodity $k \in \mathcal{K}$. The formulation of the MCFND is,

$$MCFND = \min_{y,x} \sum_{a \in \mathcal{A}} f_a y_a + \sum_{k \in \mathcal{K}} \sum_{a \in A} c_a^k x_a^k \tag{3.37}$$

$$\text{s.t} : \sum_{a \in \mathcal{A}(i)^+} x_a^k - \sum_{a \in \mathcal{A}(i)^-} x_a^k = \begin{cases} d^k & \text{if } i = O(k) \\ -d^k & \text{if } i = D(k) \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K}, \tag{3.38}$$

$$\sum_{k \in \mathcal{K}} x_a^k \leq u_a y_a \qquad \forall a \in \mathcal{A} \tag{3.39}$$

$$y \in \{0,1\}^{|\mathcal{A}|}, x \in \mathbb{R}_+^{|\mathcal{A}||\mathcal{K}|}, \tag{3.40}$$

where $\mathcal{A}(i)^+$ and $\mathcal{A}(i)^-$ are respectively the set of outward and inward arcs incident to node $i \in \mathcal{N}$. Constraints (3.38) impose the flow conservation requirements for each commodity on each node. Constraints (3.39) enforce the capacity limit on each arc and constraints (3.40) indicate the non-negativity and integrality requirements of the decision variables.

To improve the LP relaxation of the MCFND, the so-called *strong inequalities* (SIs) are often used (Chouman et al., 2017). The SIs are derived from the observation that any commodity flow on an arc cannot be larger than its corresponding demand or the arc capacity, i.e., $x_a^k \leq \min\{u_a, d^k\} y_a$. Although they are redundant at the integer points, they can remove many fractional solutions and provide a better characterization of the mixed-integer convex hull when the LP relaxation of the problem is solved.

3.5.3 Solution methods

Considering the ND problems from the solution perspective, they are NP-hard problems in all but the simplest cases (Magnanti and Wong, 1984). Generally speaking, three features in these problems can significantly change the level of resolution difficulty : (i) capacity restriction, (ii) multi-commodity flow, and (iii) side constraints. For instance, it is very well-known that for some uncapacitated network problems polynomial algorithms exist while we are unaware of any such algorithms for the capacitated counterparts. On the other hand, side constraints on the design (e.g. cyclic design or existence of multiple paths between the origin destination pairs) can significantly increase the difficulty of the problem.

In short, the main difficulty of solving the ND problems is rooted in the complex interrelation between the design and flow costs, weak LP relaxation, degeneracy, the additional challenges stemming from the very large problem dimensions characterizing most applications, and data uncertainty (Gendron et al., 1999).

Various solution algorithms have been used to solve the ND instances. This includes, for example, Lagrangian relaxation (Crainic et al., 2001; Sellmann et al., 2002; Frangioni and Gendron, 2009), cutting plane (Kliewer and Timajev, 2005; Chouman et al., 2017; Gendron and Gouveia, 2017), branch-and-cut (Günlük, 1999), Benders decomposition (Costa, 2005), heuristics (Ghamlouche et al., 2003; Hewitt et al., 2010), parallel methods (Crainic et al., 2000; Crainic and Gendreau, 2002), column generation (Maculan et al., 2002; Santini et al., 2018), etc. Among these methods, heuristics have often been the methodology of choice due to the difficulty of solving the ND problems to optimality (Gendron et al., 2018). On the other hand, significantly less efforts have been devoted to solve the stochastic variant of the ND problems due to the additional complexity stemming from the data uncertainty. In this case, the solution algorithms have mainly been based on approximations (Maggioni et al., 2016; Wang et al., 2016b) or decomposition techniques (Crainic et al., 2011; Ahmed, 2013; Crainic et al., 2016).

CHAPTER 4 ARTICLE 2: ACCELERATING THE BENDERS DECOMPOSITION METHOD: APPLICATION TO STOCHASTIC NETWORK DESIGN PROBLEMS

Ragheb Rahmaniani[1], Teodor Gabriel Crainic[2], Michel Gendreau[1], Walter Rei[2]

[1] CIRRELT & Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal H3C 3A7, Canada.

[2] CIRRELT & School of Management, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal H3C 3P8, Canada.

Abstract. This paper describes a Benders decomposition algorithm capable of efficiently solving large-scale instances of the well-known *multi-commodity capacitated network design problem* with demand uncertainty. The problem is important because it models many applications, including telecommunications, transportation, and logistics. This problem has been tackled in the literature with meta-heuristics and exact methods, but many benchmark instances, even though of moderate size, remain unsolved. To successfully apply Benders method to these instances, we propose various acceleration techniques, including the use of *cutting planes*, *partial decomposition*, *heuristics*, *stronger cuts*, *reduction* and *warm-start* strategies. Extensive computational experiments on benchmark instances were conducted to evaluate the efficiency and robustness of the algorithm as well as of the proposed strategies. The numerical results confirm the superiority of the proposed algorithm over existing ones.

We dedicate this work to Professor Jacques F. Benders who left this world January 2017.

Keywords. *Network design* ; *Stochastic programming* ; *Benders decomposition.*

History. To appear in SIAM Journal on Optimization.

4.1 Introduction

A wide range of real-world applications in diverse settings such as logistics, transportation, and telecommunications can be modeled as *Network Design* (*ND*) problems (Crainic, 2000; Minoux, 2001). In general terms, these problems revolve around the selection of a subset of (capacitated) arcs out of a potential set in order to support the flow requirements between various origin-destination (OD) pairs. The decisions are to be made such that an effective trade-off between the cost of design decisions and their impact on the performance of the resulting network is achieved, while demand and a set of problem-specific constraints are satisfied (Magnanti and Wong, 1984; Crainic et al., 2001).

Uncertainty characterizes many cases for which ND models are the methodology of choice, in particular the strategic and tactical planning processes yielding decisions to be used repetitively over a certain time horizon (Crainic et al., 2006a; Santoso et al., 2005). Moreover, the impact on the performance of a system of the uncertainty regarding its future state is far from trivial because mid to long-term decisions are generally difficult to reverse or costly to adjust (Wang et al., 2016b). *Stochastic programming* has become the methodology of choice to properly capture the uncertainty in such cases (Crainic et al., 2011).

Stochastic Network Design (*SND*) problems aim to find a single design that remains cost-effective when plausible realizations of the uncertain elements are encountered. To characterize the uncertainty set, a common practice is to use a set of discrete scenarios (Birge and Louveaux, 1997). Once the appropriate set of scenarios is generated, the SND problem can be formulated in an extensive form as a *two-stage stochastic* program. The first-stage models the design decisions, i.e., selection of arcs. The second-stage formulates the recourse actions by measuring the cost-effectiveness of the first-stage design in servicing the demands (e.g., flow decisions) for each realization of the uncertainty.

ND constitutes a challenging class of NP-hard combinatorial optimization problems. Considering the uncertainty does not make them any easier to solve as it yields much larger instances. Thus, they remain notoriously hard to address with exact methods and off-the-shelf solvers (Chouman et al., 2014; Crainic et al., 2014b). To give an idea of the inherent difficulty of the SND problems, moderate instances with more than 20 scenarios remain still unattainable for state-of-the-art algorithms and optimization solvers ; see section 4.5. Given the economical, political and environmental significance of ND problems (Crainic and Florian, 2008; Rahmaniani et al., 2014), we need efficient and accurate methodologies to address more realistically sized instances of this important family of optimization problems, particularly in stochastic settings.

Various solution algorithms have been designed to handle stochastic programs with recourse ; see (Ruszczyński, 1999) for a comprehensive review. Most of these solution procedures rely heavily on the premises of the *Benders Decomposition* (*BD*) method (Benders, 1962; Rahmaniani et al., 2017a). BD is based on a pattern of projection, outer linearization and relaxation (Geoffrion, 1970a,b). First, the model is projected onto the subspace defined by the first-stage variables. The projected term is then dualized and an equivalent model is built by enumerating all extreme points and rays of the dual polyhedron. The extreme rays indicate feasibility requirements for the first-stage variables (i.e., feasibility cuts) and the extreme points state the projected cost (i.e., optimality cuts). Enumerating all the extreme points and rays of the dual polyhedron is not computationally practical and most of the associated cuts will not be

active at an optimal solution. Therefore, a relaxation of the equivalent formulation is performed such that it initially includes no cuts, these being iteratively generated by sequentially solving a *Master Problem* (*MP*) and one or several *Subproblems* (*SPs*).

The BD method enables decomposing the SND problems according to the realization of the random elements that set the values of the uncertain parameters in the model. It thus simplifies the solution of these problems since, often, a large portion of variables and constraints are associated to the large number of scenarios used to represent the uncertainty (Boland et al., 2015). The BD algorithm is also known to be largely used as an exact method for deterministic ND problems, as it provides the means to separate the design and flow decisions (Costa, 2005). Moreover, it constitutes currently the state-of-the-art exact method for SND problems (Crainic et al., 2016). We aim to further enhance this algorithm by refining existing acceleration strategies and proposing novel ones.

We investigate, in particular, the idea of tightening the MP by means of various *valid inequalities* (*VI*s) appended in a cutting-plane fashion. More importantly, we propose a set of VIs that can be used to improve the quality of the classical Benders cuts. We also propose a new strategy to generate Pareto-optimal cuts, which is equivalent to the most widely used strategy in the literature, but numerically performs better. To avoid the generation of feasibility cuts, we propose to apply a relatively complete recourse property to the formulation that, when combined with the use of strengthened combinatorial cuts, can be tailored to enforce the original feasibility requirements of the problem. In essence, this strategy completely eliminates the need to generate feasibility cuts, in favor of a focused search for optimality cuts, which have a greater effect on improving the value of the lower bound generated by the BD method. Moreover, to mitigate the ineffective initial iterations of the BD method, we propose a strategy based on the idea of "properly deflecting the master solutions" so as to chose better dual values for the set of initial cuts. Last but not least, to handle large instances more effectively, size-reduction procedures and a simple heuristic are developed, and the algorithm is embedded in a branch-and-cut framework.

To test our method, we address the well-known *Multi-Commodity Capacitated Fixed-charge Network Design Problem with Stochastic Demands* (*MCFNDSD*). Extensive numerical experiments on a broad range of instances show that the algorithm delivers superior performance compared to state-of-the-art algorithms and solvers.

The contribution of this paper is threefold : 1) To enhance the state-of-the-art of BD algorithms, through new acceleration strategies and the refinement of existing ones ; 2) To propose an exact algorithm capable of efficiently solving benchmark instances of the MCFNDSD ; 3) To study experimentally the algorithm and proposed strategies, and to characterize their

behavior and suggest fruitful research directions.

The rest of this article is organized as follows. The problem is presented in Section 4.2, while Section 4.3 provides the literature review of the BD algorithm and its application to the problem at hand. Section 4.4 details the proposed and enhanced acceleration strategies, and Section 4.5 displays the numerical experiments and the associated analyzes. Section 4.6 gives the conclusions and future research directions.

4.2 Problem definition

In the MCFNDSD problem, a set of commodities $\mathcal{K}$ are to be simultaneously routed on a directed graph consisting a set of nodes $\mathcal{N}$ and a set of potential arcs $\mathcal{A}$. With each commodity $k \in \mathcal{K}$ is associated a stochastic demand $d^k(\omega) \geq 0$ which is to be routed from a unique origin node $O(k) \in \mathcal{N}$ to a unique destination node $D(k) \in \mathcal{N}$ for each realization ω of the uncertainty set Ω. The goal is to select a subset of the arcs to meet all demand at minimum cost, computed as the sum of the fixed design costs, charged whenever an arc is used, and the expected transportation costs. The unit transportation cost on arc $a \in \mathcal{A}$ for commodity k is denoted c_a^k, and the fixed design cost for arc $a \in \mathcal{A}$ is represented by f_a. Also, there is a capacity limit u_a on each arc $a \in \mathcal{A}$, limiting the maximum flow traversing it.

The MCFNDSD is modeled as a two-stage stochastic *mixed-integer linear program* (MILP) by using binary design variables y_a indicating if arc $a \in \mathcal{A}$ is used (1) or not (0) and a recourse function $Q : \{0,1\}^{|A|} \to \mathbb{R} \cup \{+\infty\}$ measuring the expected flow costs. The compact formulation of the MCFNDSD is

$$\text{MCFNDSD} = \min_{y \in Y} f^T y + Q(y). \tag{4.1}$$

The objective function minimizes the total fixed cost plus the expected routing costs. The feasible space of the first-stage variables, denoted by set Y, is written in compact form in order to keep the generality of formulation for a broader class of the ND problems. The recourse function associated to the design vector y is defined by $Q(y) = \mathbb{E}_\omega [\Phi(y;\omega)]$, where the function $\Phi(y;\omega)$ can be defined by using continuous flow variables $x_a^k(\omega) \geq 0$ to reflect the amount of flow on arc $a \in \mathcal{A}$ for commodity $k \in \mathcal{K}$ under realization $\omega \in \Omega$. The recourse

problem $\omega \in \Omega$ is

$$\Phi(y;\omega) = \min_{x(\omega)\in\mathbb{R}_+^{|\mathcal{A}||\mathcal{K}|}} \sum_{k\in\mathcal{K}}\sum_{a\in\mathcal{A}} c_a^k x_a^k(\omega) \tag{4.2}$$

$$\text{s.t.} \sum_{a\in\mathcal{A}(i)^+} x_a^k(\omega) - \sum_{a\in\mathcal{A}(i)^-} x_a^k(\omega) = d_i^k(\omega) \qquad \forall i \in \mathcal{N}, \forall k \in \mathcal{K} \tag{4.3}$$

$$\sum_{k\in\mathcal{K}} x_a^k(\omega) \leq u_a y_a \qquad \forall a \in \mathcal{A}, \tag{4.4}$$

where $A(i)^+$ and $A(i)^-$ indicate the set of outward and inward arcs incident to node i. The vectors $d_i^k(\omega)$ express the nodal balance and are defined as

$$d_i^k(\omega) = \begin{cases} d^k(\omega) & \text{if } i = O(k) \\ -d^k(\omega) & \text{if } i = D(k) \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

The objective function (4.2) minimizes the total flow costs. Constraint set (4.3) imposes the flow conservation requirements for each commodity on each node. Constraints (4.4) enforce the capacity limit on each arc.

4.2.1 A brief review on the MCFNDSD

The outlined MCFNDSD problem is notoriously difficult to solve. Its main difficulty is rooted in the complex interrelation between the design and flow costs, weak linear programming (LP) relaxation, degeneracy, and the additional challenges stemming from the very large problem dimensions characterizing most applications (Gendron et al., 1999). Therefore, heuristics have been the most used solution algorithms to obtain good solutions within reasonable running times. Crainic et al. (2011) devised a master-slave parallel progressive hedging based meta-heuristic. They decomposed the problem according to the scenarios where each SP is an NP-hard deterministic ND formulation. Then, the algorithm seeks a consensus for the design decisions among the SPs by systematically updating the fixed costs. An improvement to this algorithm, by investigating the idea of having multiple subproblems, is proposed in Crainic et al. (2014b). The authors observed that despite the increased difficulty of each subproblem, more elegant results could be obtained. A BD-based heuristic for the problem with large number of scenarios was introduced in Boland et al. (2015), the authors reporting encouraging results compared to the classical algorithm and state-of-the-art of commercial solvers.

It has been only quite recently that tight bounds were obtained for the deterministic version

of the problem at hand (Chouman et al., 2014). The results obtained provide a promising overture toward handling deterministic ND problems exactly. However, it is not a viable tool to address the stochastic variant of the problem, since it considers all the scenarios at the same time. An enhanced BD method and applied to the SND problems was presented in Crainic et al. (2014a) , the authors further improved the algorithm in Crainic et al. (2016). To the best of our knowledge, it constitutes the state-of-the-art of the exact methods for the MCFNDSD problem. Due to the closeness of this algorithm to the method presented in this article, we shall shortly discuss it in more details.

4.3 The Benders decomposition method

Let $\bar{y}$ be an arbitrary value of y, π and α be the dual variables of (4.3) and (4.4). The dual of $\Phi(\bar{y};\omega)$ can be stated as

$$\text{DSP}(\bar{y};\omega) := \max_{\pi\in\mathbb{R}^{|\mathcal{N}||\mathcal{K}|},\alpha\in\mathbb{R}_+^{|\mathcal{A}|},s\in\mathbb{R}_+^{|\mathcal{A}||\mathcal{K}|}} \sum_{k\in\mathcal{K}} d^k(\omega)\left(\pi^k_{O(k)} - \pi^k_{D(k)}\right) - \sum_{a\in\mathcal{A}} u_a\bar{y}_a\alpha_a \tag{4.6}$$

$$\text{s.t.} \quad \pi^k_{a^-} - \pi^k_{a^+} - \alpha_a + s^k_a = c^k_a \qquad \forall a\in\mathcal{A}, k\in\mathcal{K}, \tag{4.7}$$

where a^+ and a^- respectively indicate the tail and head of arc a, and s^k_a is the slack variable associated to the constraint (4.7) for commodity $k\in\mathcal{K}$ and arc $a\in\mathcal{A}$. The $DSP(\bar{y};\omega)$ can be either unbounded or feasible, if the dual polyhedron is not empty. The former case indicates the infeasibility of the $\bar{y}$ solution. Thus, there is direction of unboundedness which satisfies

$$\sum_{k\in\mathcal{K}} d^k(\omega)\left(\pi^k_{O(k)} - \pi^k_{D(k)}\right) - \sum_{a\in\mathcal{A}} u_a\bar{y}_a\alpha_a > 0, \tag{4.8}$$

the (π,α) is the solution of the dual cone obtained by replacing c^k_a with 0 in constraint (4.7). In the latter case, capturing the optimal value of $DSP(\bar{y};\omega)$ in a single variable $\theta_\omega\in\mathbb{R}^1$, the goal is being θ_ω never exceeds the optimal value associated to $\bar{y}$, i.e.,

$$\sum_{k\in\mathcal{K}} d^k(\omega)\left(\pi^k_{O(k)} - \pi^k_{D(k)}\right) - \sum_{a\in\mathcal{A}} u_a\bar{y}_a\alpha_a > \theta_\omega. \tag{4.9}$$

The BD method exploits these results. It iteratively solves an MIP problem, called MP, which involves only the y and θ variables. The MP has initially no constraints except for those imposed on the design variables. Each time the MP is solved, the optimal value of the design variables is extracted and fixed in the formulation (4.6)-(4.7) for each $\omega\in\Omega$. A feasibility or optimality cut for each scenario can then be generated by complementing (4.8) or (4.9) and replacing the fixed design variables with arbitrary values. These cuts are then

inserted in the MP and the next iteration is performed. Accordingly, the MP at iteration t has the following form

$$MP^t = \min_{y \in Y, \theta \in \mathbb{R}^{|\Omega|}} f^T y + \mathbb{E}_\omega \left[\theta_\omega\right] \tag{4.10}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{K}} d^k(\omega) \left(\bar{\pi}^k_{O(k)} - \bar{\pi}^k_{D(k)}\right) - \sum_{a \in \mathcal{A}} u_a \bar{\alpha}_a y_a \leq 0 \quad \forall \omega \in \Omega, (\bar{\pi}, \bar{\alpha}) \in E^t_\omega \tag{4.11}$$

$$\sum_{k \in \mathcal{K}} d^k(\omega) \left(\bar{\pi}^k_{O(k)} - \bar{\pi}^k_{D(k)}\right) - \sum_{a \in \mathcal{A}} u_a \bar{\alpha}_a y_a \leq \theta_\omega \quad \forall \omega \in \Omega, (\bar{\pi}, \bar{\alpha}) \in L^t_\omega, \tag{4.12}$$

where E^t_ω and L^t_ω stand for the set of feasibility and optimality cuts associated to scenario ω up to iteration t. This process continues until the upper and lower bounds coincide, To verify the optimality gap, the MP gives a valid lower bound (LB) on the optimal cost because it is a relaxation of the Benders equivalent formulation. On the other hand, the cost of the $\bar{y}$ solution plus the expected cost of the SPs gives a valid upper bound (UB) on the optimal cost, because it is equivalent to fixing the current solution $\bar{y}$ in the original formulation.

4.3.1 Literature review on acceleration strategies

The BD method is closely related to other decomposition methods such as Dantzig-Wolfe and Lagrangian decomposition, see e.g., Rahmaniani et al. (2017a) for more information. It has proven successful for a wide range of difficult optimization problems, including ND problems (Costa, 2005). A naive implementation of the classical algorithm may however preform disappointingly. Researchers have thus explored ways to overcome the drawbacks of the algorithm when applied to various optimization problems. We briefly review the acceleration strategies that are closest to what we will present in this article and refer the interested readers to Rahmaniani et al. (2017a) for a detailed review on this subject.

Probably the most studied line of accelerations revolves around the generation of cuts. Magnanti and Wong (1981) were the first to propose the generation of non-dominated optimality cuts when the dual SP has multiple optima for a given feasible solution. This strategy, which entails the solution of an additional SP, is instantiated using a core point of the MP to identify a non-dominated cut. Papadakos (2008) emphasized that the previous method can be inefficient due to the dependency on the auxiliary SP and the difficulty of extracting the core point, and showed how alternative core points can be used in each iteration to generate Pareto-optimal cuts by means of an independent SP. Sherali and Lunday (2013) showed that Pareto-optimal cuts could be generated by perturbing the right-hand side values of the constraints in the primal SP. Alternative feasibility cuts for binary problems where the SP is a feasibility program and involves big-M constraints, were proposed in Codato and Fi-

schetti (2006). The authors showed that the combinatorial cuts can efficiently be separated by searching for minimal infeasible subsystems in the solutions of the relaxed MP. Bodur et al. (2017) employed Gomory mixed-integer cuts to strengthen the classical Benders cuts, by adding them to the SP each time it is solved.

The numerical burden of iteratively solving an MIP MP and a series of SPs is a major drawback of the BD method. Geoffrion and Graves (1974) proposed to solve the MP each time to ϵ-optimality and steadily decrease the ϵ value as the algorithm proceeds in order to ensure the global convergence. Similar idea at the SP level has been applied in Zakeri et al. (2000) to extract sub-optimal solutions of the dual polyhedron. McDaniel and Devine (1977) showed that valid cuts can be produced when the LP relaxation of the MP is solved. The authors proposed to apply the algorithm in two phases. In the first phase, all integrality requirements are ignored to quickly generate cuts and tighten the master formulation. In the second phase, the integrality constraints are reintroduced and the solution process continues until the global optimum is found. Solving the MP heuristically has been proposed by Côté and Laughton (1984). The authors applied Lagrangian relaxation on the optimality and feasibility cuts to maintain the nice structure of their MP. Related to this general idea, different (meta-)heuristic algorithms have been employed to faster optimize the MP and avoid many costly iterations of the MP (Poojari and Beasley, 2009). Similarly, heuristics have been employed to extract approximate optimality (Raidl et al., 2014) and feasibility cuts (Luong, 2015). To avoid solving a MILP program at each iteration, modern implementations of the BD method work by building a single branch-and-bound tree (e.g., Gendron et al. (2016)). The algorithm generates the Benders cuts for the integer (and possibly fractional) solutions encountered inside the tree, and guarantees the convergence. Finally, managing the size of the MP by periodically removing ineffective cuts has appeared as a promising technique to reduce the computational burden, especially when multiple cuts per iteration are added to the MP (e.g., Pacqueau et al. (2012)).

Enhancing the quality of the master solutions, particularly at the initial iterations, has been the subject of various researches. The instability of the MP in respect to the generated solutions is one of the main drawbacks of the BD which can cause slow convergence due to initial large steps and excessive oscillations when it approaches a local optimum. A constraint to restrict the Hamming distance of the generated solutions from a stabilizing point was proposed to address the issue in Santoso et al. (2005). Heuristics are also often used as a warm-start strategy to generate a set of high-quality initial solutions and their associated cuts to tighten the relaxed MP, e.g., Lin and Üster (2014) or exploring the neighborhoods of the current solution, e.g., Rei et al. (2009). Strengthening the relaxed MP with VIs is also a widely-used strategy to tighten the initial MP, e.g., Saharidis et al. (2011). Birge and

Louveaux (1988) demonstrated that it is preferable to add a single cut per SP when the decomposition yields several independent SPs.

The last category of accelerations is specifically tailored for two-stage stochastic programs. This idea was first tested in MirHassani et al. (2000) where the authors proposed to retain in the MP a scenario SP associated to the maximum demand. Crainic et al. (2014a) proposed various strategies to properly chose and retain a subset of the SPs in the MP, and then significantly improvements to this strategy through simultaneously selecting and creating SPs to be retained in the MP (Crainic et al., 2016) . The authors referred to this technique as partial-decomposition strategy (PDS).

In short, not a single strategy is a silver bullet and various accelerations need to be incorporated into the BD framework. This is particularly true for realistic instances of SND problems, for which multi-cut reformulation, Pareto-optimal cut generation scheme, a two phase approach, single search tree strategy and PDS are considered. As the results of this paper indicate, however, these strategies are not sufficient to reach a high-performance BD method. Our goal is thus to contribute to the enhancement of the categories mentioned above, as outlined in the following section.

4.4 Enhancing Benders decomposition method

We present in this part various strategies to address several drawbacks of the algorithm, including : (1) weak relaxation of the MP, (2) generation of low-quality cuts, (3) ineffective initial iterations, (4) slow progression of the primal bound, and (5) time consuming iterations. In section 4.4.1, VIs are proposed to alleviate the weak relaxation of the MP. The cut-generation scheme is revisited in section 4.4.2 to efficiently generate stronger/better cuts. A warm-start strategy is presented in section 4.4.3 to improve the quality of the initial cuts. Finally, reduction strategies, heuristic, branching and node selection rules are discussed in section 4.4.4 to effectively handle the integer phase of the algorithm.

4.4.1 Valid inequalities

At the relaxation step of the BD method, an important part of the formulation is projected out. Thus, the algorithm exhibits a poor performance due to the absence of sufficient leading information. Although the PDS can considerably dampen these undesirable behaviors, more significant accelerations can be achieved by making use of strong VIs. We shall numerically show that the best results can be attained when both strategies are applied. Note that the benefit of VIs is well established in the literature. To the best of our knowledge, they are

merely used as a warm-start to the MP. We propose a cutting plane that extends the use of the VIs throughout the whole solution process. We observed that better performance can be achieved when these VIs are prioritized over the Benders cuts.

Lower bounding function

The first class of VIs estimates the projected terms of the objective function. We observe that, for any OD pair, a minimum flow cost can be calculated since the corresponding demand must be satisfied. Let $P_{O(k)D(k)}$ indicate the shortest path connecting the origin of commodity k to its destination. Inequality (4.13) provides a lower estimation on the recourse cost of the scenario ω. This inequality can be justified based on the simple argument that the routing cost for each commodity is at least as large as that of the cheapest route satisfying it.

$$\theta_\omega \geq \sum_{k \in \mathcal{K}} \sum_{a \in P_{O(k)D(k)}} d_k(\omega) c_a^k \qquad \forall \omega \in \Omega. \tag{4.13}$$

Inequality (4.13) provides a weak estimation of the recourse cost, however. This is because the considered problem is capacitated. More importantly, it gives no leading information on the first-stage variables. To provide a stronger bound, referred to as the *lower bounding function* (*LBF*), we make use of a deterministic *Multi-Commodity Capacitated Network Flow* (*MCNF*) problem with minimum demands, i.e., $d_k = min_{\omega \in \Omega}\{d_k(\omega)\}, \forall k \in \mathcal{K}$; see Appendix A for the formulation. This problem is equivalent to the LP relaxation of the deterministic version of the considered ND problem with the demand $d_k, \forall k \in \mathcal{K}$. Its solution provides a valid lower bound on the cost of each SP, since the demands are replaced by the minimum volumes.

Theorem 4.1. *Let $\bar{x}_a^k$ and $\bar{y}$ be the optimal solution of the MCNF problem, $\bar{\pi}$ be the dual values of the flow conservation constraints, then (4.14) is a VI for the MP.*

$$\theta_\omega \geq \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} c_a^k \bar{x}_a^k + \sum_{k \in \mathcal{K}} \left(d_k(\omega) - d_k\right) \left(\bar{\pi}_{O(k)}^k - \bar{\pi}_{D(k)}^k\right) + \sum_{a \in \mathcal{A}} f_a \left(\bar{y}_a - y_a\right) \ \forall \omega \in \Omega. \tag{4.14}$$

Proof. See Appendix A. □

Cutset inequalities

Based on the feasibility requirements of the problem at hand, there should be sufficient capacity installed across any partition of the network to support the commodity flows. Let

$\bar{N} \subset \mathcal{N}$ be any nonempty subset of the node set $\mathcal{N}$ and $\underline{N}$ its complement, i.e., $\underline{N} = N \backslash \bar{N}$, and the corresponding cutset $(\bar{N}, \underline{N}) = \{a \in \mathcal{A} | a^+ \in \bar{N}, a^- \in \underline{N}\}$. Let $K(\bar{N}, \underline{N}) = \{k \in \mathcal{K} | O(k) \in \bar{N}, D(k) \in \underline{N}\}$ be the associated commodity subset. We define the maximum flow over this cutset as $d^{max}_{(\bar{N},\underline{N})} = max_{\omega \in \Omega}\{\sum_{k \in \mathcal{K}(\bar{N},\underline{N})} d_k(\omega)\}$, where $(\bar{N}, \underline{N})$ is a valid cutset if $d^{max}_{(\bar{N},\underline{N})} > 0$. We can derive *cover inequalities* (CIs) using definition 4.1.

Definition 4.1. *$C \subseteq (\bar{N}, \underline{N})$ is a cover if the total capacity of the arcs in $(\bar{N}, \underline{N}) \backslash C$ does not support (cover) the flow of demand, i.e., $\sum_{a \in (\bar{N},\underline{N}) \backslash C} u_a < d^{max}_{(\bar{N},\underline{N})}$; and a cover set C is* minimal *if opening any arc in C is sufficient to cover the demand, i.e., $\sum_{a \in (\bar{N},\underline{N}) \backslash C} u_a + u_q \geq d^{max}_{(\bar{N},\underline{N})}, \forall q \in C$.*

For any cover set $C \subseteq (\bar{N}, \underline{N})$, the CI takes the form $\sum_{a \in C} y_a \geq 1$, imposing the necessity of opening at least one arc in the set C to meet the flow requirements.

Minimum cardinality inequalities (*MCIs*) can also be derived for the cutset $(\bar{N}, \underline{N})$, indicating the least number of arcs, denoted $l_{(\bar{N},\underline{N})}$), that must be opened in any feasible solution. Let the arc capacities in $(\bar{N}, \underline{N})$ be sorted and numbered in a non-increasing order, i.e., $u_a \geq u_{a+1}, a \in (\bar{N}, \underline{N}), a = 1, ..., |(\bar{N}, \underline{N})|$. We define $l_{(\bar{N},\underline{N})} = max\{h | \sum_{t=1,...,h} u_t < d^{max}_{(\bar{N},\underline{N})}\} + 1$. Then, $\sum_{a \in (\bar{N},\underline{N})} y_a \geq l_{(\bar{N},\underline{N})}$ is a valid MCI cut.

To separate and lift the cutset inequalities, we follow the procedure proposed by Chouman et al. (2017). Generating good cutsets quickly is of critical importance for the effectiveness of this family of inequalities. We have implemented the same cutset generation scheme as Chouman et al. (2017) with cardinality of one and two because we observed that they are responsible for most of the LB improvement in our algorithm.

Network connectivity cuts

The feasibility requirements of the ND problem requires the existence of at least one connecting path with sufficient capacity between each OD pair. This implies that a sufficient number of arcs exiting (entering) each origin (destination) node should be opened to provide the capacity required to move the demand out of (into) the respective node. Additionally, at least one arc should enter and one should exit any node that might be a transshipment node for a commodity. These conditions are generally enforced through constraints (4.3). Yet, after the decomposition, the enforcing constraints at each node are eliminated and the network is disconnected, particularly for initial iterations of the algorithm. To alleviate this issue, we propose the following *network connectivity inequalities* (*NCIs*).

Consider node $i \in \mathcal{N}$ with $d^{max}_i = max_{\omega \in \Omega}\{\sum_{k \in \mathcal{K} | i = O(k)} d_k(\omega)\}$ (the inner sum computed on $D(k) = i$ for destination nodes), and set $\mathcal{A}^+(i)$ ($\mathcal{A}^-(i)$) of exiting (entering) arcs. If $d^{max}_i > 0$,

sort (and number) the arcs in $\mathcal{A}^+(i)$ in non-increasing order, i.e., $u_a \geq u_{a+1}, a \in \mathcal{A}^+(i), a = 1, ..., |\mathcal{A}^+(i)|$ (same for $\mathcal{A}^-(i)$). Then, the least number of exiting (entering) arcs that must be opened in any feasible solution is $l_i^+ = max\{h | \sum_{t=1,...,h} u_t < d_i^{max}\} + 1$, yielding the following valid cardinality NCI, $\sum_{a \in \mathcal{A}^+(i)} y_a \geq l_i^+$.

Let $\mathcal{I} = \{i \in \mathcal{N} | i \neq O(k) \text{ and } i \neq D(k), \forall k \in \mathcal{K}\}$ be the set of purely intermediary nodes (i.e., no commodity originates or terminates at such a node). The following proposition defines a set of NCIs addressing the connectivity issue.

Proposition 4.1. *Given the set of intermediary nodes $\mathcal{I}$, the following set of inequalities are valid for the MCFNDSD and thus for the MP.*

$$y_a \leq \sum_{b \in \mathcal{A}^+(i)} y_b \quad \forall i \in \mathcal{I}, a \in \mathcal{A}^-(i), \tag{4.15}$$

$$y_b \leq \sum_{a \in \mathcal{A}^-(i)} y_a \quad \forall i \in \mathcal{I}, b \in \mathcal{A}^+(i). \tag{4.16}$$

Proof. For an intermediary node, exiting arcs can exist if there is at least one entering arc, and vice versa. This follows from the positive fixed costs and the flow conservation requirements. □

The inequalities (4.15) and (4.16) can be further strengthened for the first phase of the algorithm according to the following lemma.

Lemma 4.1. *If the integrality requirement is relaxed, i.e., $y_a \in [0, 1], \forall a \in \mathcal{A}^+(i) \cup A^-(i)$ and $i \in \mathcal{I}$, then inequalities (4.15) and (4.16) can be rewritten as*

$$\sum_{a \in \mathcal{A}^+(i)} u_a y_a - \sum_{a \in \mathcal{A}^-(i)} u_a y_a = 0 \qquad \forall i \in \mathcal{I}. \tag{4.17}$$

Proof. The inward flow to node $i \in \mathcal{I}$ is equal to $max_{\omega \in \Omega}\{\sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}^+(i)} x_a^k(\omega)\}$ which entails $\sum_{a \in \mathcal{A}^+(i)} u_a y_a \geq max_{\omega \in \Omega}\{\sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}^+(i)} x_a^k(\omega)\}$ at any feasible solution. Since node i is an intermediary node, the same amount of flow should exit from it, i.e., $max_{\omega \in \Omega}\{\sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}^+(i)} x_a^k(\omega)\} = max_{\omega \in \Omega}\{\sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}^-(i)} x_a^k(\omega)\}$, which gives $\sum_{a \in \mathcal{A}^-(i)} u_a y_a \geq max_{\omega \in \Omega}\{\sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}^+(i)} x_a^k(\omega)\}$. When the binary requirement is relaxed, the flow approximates the exact amount of the opened capacity. This completes the proof since $\sum_{a \in \mathcal{A}^+(i)} u_a y_a = max_{\omega \in \Omega} \{\sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}^+(i)} x_a^k(\omega)\} = max_{\omega \in \Omega}\{\sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}^-(i)} x_a^k(\omega)\} = \sum_{a \in \mathcal{A}^-(i)} u_a y_a$. □

Flow cuts

The PDS provides the possibility of generating VIs involving both flow and design variables. *Strong inequalities* (SIs) are the well-known examples of such inequalities. SIs are very effective when incorporated in a cutting-plane method. They are derived from the observation that any commodity flow on an arc cannot be larger than its corresponding demand or the arc capacity. Let $\bar{\Omega}$ indicate the set of retained and created scenarios and $b_a^k(\omega) = min\{d_k(\omega), u_a\}$, then

$$x_a^k(\omega) \leq b_a^k(\omega) y_a \qquad \forall \omega \in \bar{\Omega},\, a \in \mathcal{A},\, k \in \mathcal{K}, \tag{4.18}$$

is valid for the MCFNDSD and hence for the MP. SIs entail an easy separation and are added to the MP if they violate the current solution by at least η units. We also examined flow cover and flow pack inequalities (see Chouman et al. (2017)). We decided, however, to exclude them from our algorithm because their contribution on the LB improvement tended to be limited when the above inequalities were already appended.

4.4.2 Strengthening the Benders cuts

After focusing on improving the quality of the y solutions, we now turn to the cut-generation scheme. We reinterpret Magnanti and Wong (1981)'s strategy in order to generate Pareto-optimal cuts more efficiently, propose an alternative strategy to avoid generation of feasibility cuts, and further improve the quality of the optimality cuts by making use of VIs.

Generating Pareto-optimal cuts

The primal SP is degenerate, thus, multiple optimal solutions can be extracted from its dual formulation. Each alternative solution produces an optimality cut of particular strength. The dual values thus need to be judiciously selected in order to append the strongest cuts to the MP. Magnanti and Wong (1981) employed the notion of *dominance* to generate such cuts.

Definition 4.2. *Let $U(DSP(\bar{y};\omega))$ be the polyhedron of $DSP(\bar{y};\omega)$. An optimality cut (4.12) corresponding to $(\hat{\pi},\hat{\alpha}) \in U(DSP(\bar{y};\omega))$, dominates or is stronger than that corresponding to $(\bar{\pi},\bar{\alpha}) \in U(DSP(\bar{y};\omega))$ if $\sum_{k\in\mathcal{K}} d^k(\omega)(\hat{\pi}^k_{O(k)} - \hat{\pi}^k_{D(k)}) - \sum_{a\in\mathcal{A}} u_a\hat{\alpha}_a y_a \geq \sum_{k\in\mathcal{K}} d^k(\omega)(\bar{\pi}^k_{O(k)} - \bar{\pi}^k_{D(k)}) - \sum_{a\in\mathcal{A}} u_a\bar{\alpha}_a y_a$ for all $y \in Y$ with strict inequality for at least one point $y \in Y$ and $(\hat{\pi};\hat{\alpha}) \in U(DSP(\bar{y},\omega))$ is called Pareto-optimal if it is not dominated by any other cut.*

The authors used the *core point* notion to formulate an auxiliary SP to extract the Pareto-optimal cut.

Definition 4.3. *Let y^0 be a core-point laying in the relative interior of the convex hull of the subregion defined by the MP variables and $\overline{DSP}(\bar{y};\omega)$ indicate the objective value of the dual SP (4.6–4.7). Then, the optimal solution $(\hat{\pi},\hat{\alpha})$ of the following program is Pareto-optimal :*

$$DSP(y^0,\omega) = \max_{\pi\in\mathbb{R}^{|\mathcal{N}||\mathcal{K}|},\alpha\in\mathbb{R}_+^{|\mathcal{A}|},s\in\mathbb{R}_+^{|\mathcal{A}||\mathcal{K}|}} \sum_{k\in\mathcal{K}} d^k(\omega)\left(\pi^k_{O(k)}-\pi^k_{D(k)}\right) - \sum_{a\in\mathcal{A}} u_a y^0_a \alpha_a \tag{4.19}$$

$$s.t. \qquad \pi^k_{a^-} - \pi^k_{a^+} - \alpha_a + s^k_a = c^k_a \qquad \forall a\in\mathcal{A}, k\in\mathcal{K} \tag{4.20}$$

$$\sum_{k\in\mathcal{K}} d^k(\omega)\left(\pi^k_{O(k)}-\pi^k_{D(k)}\right) - \sum_{a\in\mathcal{A}} u_a \bar{y}_a \alpha_a = \overline{DSP}(\bar{y};\omega), \tag{4.21}$$

This problem is solved after solving the regular SP (4.6)–(4.7) in order to derive the corresponding Pareto-optimal cut. The dependency of this approach on the auxiliary SP (4.19)–(4.21) can, however, overshadow algorithm's performance. It doubles the number of the SPs and the secondary problem is relatively more difficult to solve. We found that the latter issue is to a very large extent due to the presence of the equality constraint (4.21), which may entail numerical instabilities as well.

The purpose of the equality constraint (4.21) is to restrict the SP (4.6)-(4.7) to the optimal face of the dual polyhedron where all the alternate optimal solutions exist. The objective function (4.19) attempts to pick the one, among the available alternatives, which gives the tightest cut as measured from an interior point of the MP. This constraint is not required, however, to extract such solution. To remove it from formulation (4.19)-(4.21), we recall from linear programming theory that an alternate optimal solution exists if at least one nonbasic variable possesses a reduced cost of zero. Thus, when we identify that at least one alternate dual optima exists, we search the best one to generate the cut by restricting the SP to the optimal face. To do so, we acknowledge that variables with nonzero reduced cost must maintain their current value in any alternate optimal solution. Moreover, from the duality theory, an active constraint on the optimal face has a nonzero dual value. Thus, each inequality constraint with nonzero dual value should be converted into an equality. This is equivalent to fixing the slack variables with nonzero reduced cost to zero. Using these definitions, Lemma 4.2 formally introduces an equivalent SP to extract Pareto-optimal cuts.

Lemma 4.2. *Let $(\vec{\alpha},\vec{\pi},\vec{s})$ indicate the vector of variables with nonzero reduced cost, as ob-*

tained from solving the $DSP(\bar{y};\omega)$ problem. Then, the solution of

$$DSP(y^o;\omega)= \max_{\pi\in\mathbb{R}^{|\mathcal{N}||\mathcal{K}|},\alpha\in\mathbb{R}_+^{|\mathcal{A}|},s\in\mathbb{R}_+^{|\mathcal{A}||\mathcal{K}|}} \sum_{k\in\mathcal{K}} d^k(\omega)(\pi^k_{O(k)} - \pi^k_{D(k)}) - \sum_{a\in\mathcal{A}} u_a y^0_a \alpha_a \tag{4.22}$$

$$s.t. \qquad \pi^k_{a^-} - \pi^k_{a^+} - \alpha_a + s^k_a = c^k_a \qquad \forall a\in\mathcal{A}, k\in\mathcal{K} \tag{4.23}$$

$$(\vec{\alpha},\vec{\pi},\vec{s}) = \mathbf{0}, \tag{4.24}$$

is equivalent to that obtained from the Magnanti-Wong problem.

Proof. The equivalence is evident because only variables with zero reduced cost can change their value, which does not affect the objective value of (4.6)-(4.7). Note that, the variables are unbounded and if they have a nonzero reduced cost, their value at the optimal solution is necessarily zero. □

The equality (4.24) fixes a set of variables to their current value (zero), which can efficiently be handled by the optimization solvers, e.g., CPLEX. This variable fixation also helps the solver to further reduce the scale of the auxiliary SP (4.22)-(4.24) through inspection and removal of the unnecessary constraints and variables.

Improving the optimality cuts

We make use of VIs to improve the quality of the optimality cuts as well as the LP relaxation. The underlying idea is to apply a cutting-plane method on each SP to tighten its formulation before generating the Benders cut. To do so, once the SP is solved, we extract the value of the flow variables, denoted $\bar{x}$. Given the current $(\bar{x},\bar{y})$ solution, violated VIs can be extracted and added to the SP. This is equivalent to adding columns to the dual SP. We show that the tightened SP yields stronger cuts, which also improves the LP bound.

Various VIs can be used to realize the aforementioned goal. We have experimented with two well-known families of inequalities, namely SIs and *flow pack* (FP) inequalities (Chouman et al., 2017). In the rest of this section, we present the theoretical results only for the case when the SIs are used. These results can be easily generalized for other VIs.

To demonstrate the impact of the SIs on the optimality cuts, assume that they are added to the SP (4.2)–(4.4) and β indicates the associated dual variables. The dual formulation,

referred to as *Strong Dual Subproblem* (SDSP), is

$$SDSP(\bar{y};\omega) = \max \sum_{k\in\mathcal{K}} d^k(\omega)(\pi^k_{O(k)} - \pi^k_{D(k)}) - \sum_{a\in\mathcal{A}} \bar{y}_a(u_a\alpha_a - \sum_{k\in\mathcal{K}} b^k_a(\omega)\beta^k_a) \tag{4.25}$$

$$\text{s.t.} \quad \pi \in \mathbb{R}^{|\mathcal{N}||\mathcal{K}|}, \alpha \in \mathbb{R}^{|\mathcal{A}|}_+, s, \beta \in \mathbb{R}^{|\mathcal{A}||\mathcal{K}|}_+ \tag{4.26}$$

$$\pi^k_{a^-} - \pi^k_{a^+} - \alpha_a - \beta^k_a + s^k_a = c^k_a \qquad \forall a \in \mathcal{A}, \quad k \in \mathcal{K}. \tag{4.27}$$

In comparison to problem (4.6)–(4.7), the above formulation is larger in scale and may entail a higher degree of degeneracy. This is, however, well compensated through significant reductions in the number of iterations. Proposition 4.2 establishes the relation among the two dual formulations $SDSP(\bar{y};\omega)$ and $DSP(\bar{y};\omega)$.

Proposition 4.2. *For a given first-stage solution $\bar{y}$, the relation $SDSP(\bar{y};\omega) \geq DSP(\bar{y};\omega)$ among the two formulations of the SPs always holds true.*

Proof. Consider the primal form of the two dual programs (4.6)-(4.7) and (4.25)-(4.27), respectively denoted $\Phi(\bar{y};\omega)$ and $\bar{\Phi}(\bar{y};\omega)$. The objective function terms in both problems are equal. The solution space of the latter is, however, a subset of the former due to the additional constraints. This implies that the latter is a relaxation of the former : $\bar{\Phi}(\bar{y};\omega) \geq \Phi(\bar{y};\omega)$. According to the strong duality theory, we have $SDSP(\bar{y};\omega) = \bar{\Phi}(\bar{y};\omega) \geq \Phi(\bar{y};\omega) = DSP(\bar{y};\omega)$. □

We can now formally state the effect of the SIs on the optimality cuts.

Proposition 4.3. *The cut generated from $SDSP(\bar{y};\omega)$ is not dominated by the one extracted from $DSP(\bar{y};\omega)$.*

Proof. To prove the claim by contradiction, assume that $(\bar{\pi}_1, \bar{\alpha}_1, \bar{\beta}_1) \in U(DSP(\bar{y};\omega))$ dominates $(\bar{\pi}_2, \bar{\alpha}_2, \bar{\beta}_2) \in U(SDSP(\bar{y};\omega))$, where $\beta_1 = 0$. (Note, to lighten the presentation, we present the equations using a vector representation.) Based on Definition 4.2, we have

$$d^T\bar{\pi}_1 - \bar{\alpha}_1^T uy - b^T\bar{\beta}_1 y \geq d^T\bar{\pi}_2 - \bar{\alpha}_2^T uy - b^T\bar{\beta}_2 y \qquad \forall y \in Y, \tag{4.28}$$

such that there is at least one $\bar{y} \in Y$ that

$$d^T\bar{\pi}_1 - \bar{\alpha}_1^T u\bar{y} - b^T\bar{\beta}_1\bar{y} > d^T\bar{\pi}_2 - \bar{\alpha}_2^T u\bar{y} - b^T\bar{\beta}_2\bar{y}. \tag{4.29}$$

On the other hand, based on inequality (4.28) we have

$$d^T\bar{\pi}_1 - \bar{\alpha}_1^T uy^0 - b^T\bar{\beta}_1 y^0 \geq d^T\bar{\pi}_2 - \bar{\alpha}_2^T uy^0 - b^T\bar{\beta}_2 y^0 \qquad \forall y^0 \in Y^c, \tag{4.30}$$

where Y^c is convex hull of Y. The last inequality is true because any $y^0 \in Y^c$ can be expressed as $y^0 = \sum_{t\in T} \lambda_t y_t$, $\sum_{t\in T} \lambda_t = 1$, $\lambda_t \geq 0$ for some finite number of $y_1, ..., y_{|T|} \in Y$. On the other hand, we know that for y^0 the following inequality holds according to proposition 4.2

$$d^T\bar{\pi}_2 - \bar{\alpha}_2^T u y^0 - b^T\bar{\beta}_2 y^0 \geq d^T\bar{\pi}_1 - \bar{\alpha}_1^T u y^0 - b^T\bar{\beta}_1 y^0. \tag{4.31}$$

Note $y^0 \in ri\,(Y^c)$ because it is a Magnanti-Wong point. Using (4.30) and (4.31), we have

$$d^T\bar{\pi}_1 - \bar{\alpha}_1^T u y^0 - b^T\bar{\beta}_1 y^0 = d^T\bar{\pi}_2 - \bar{\alpha}_2^T u y^0 - b^T\bar{\beta}_2 y^0. \tag{4.32}$$

Based on Theorem 6.4 of Rockafellar (1970), it exists a scalar $\theta > 1$ such that $\hat{y} = \theta y^0 + (1-\theta)\bar{y}$ belongs to Y^c. Multiplying equality (4.32) by θ and the strict inequality (4.29) by $1-\theta$ (which reverses the inequality), and adding these two, we get the following relation

$$d^T\bar{\pi}_1 - \bar{\alpha}_1^T u\hat{y} = d^T\bar{\pi}_1 - \bar{\alpha}_1^T u\hat{y} - b^T\bar{\beta}_1\hat{y} < d^T\bar{\pi}_2 - \bar{\alpha}_2^T u\hat{y} - b^T\bar{\beta}_2\hat{y}, \tag{4.33}$$

which contradicts inequality (4.28) and hence our assumption. □

We will numerically verify in Section 4.5.2 that applying a cutting plane on each SP yields considerable improvements in terms of iterations and LP bounds.

Feasibility cuts

As defined in constraints (4.11), the feasibility cuts do not improve the values of the θ variables. They also make the solution of the MP more complicated. More importantly, we observed that, in most iterations, the infeasibility of the $\bar{y}$ solution is rather limited, meaning that merely a small portion of the demands is left unsatisfied. In such cases, optimality cuts can still be generated instead of feasibility cuts which, considering our preliminary results, is a more numerically efficient strategy. To do so, one can employ the strategy proposed in Saharidis and Ierapetritou (2010) solving an auxiliary MILP problem in order to relax a minimum number of constraints of the SP so that an optimality cut can be generated. We propose a different and simpler strategy, however, which avoids solving an auxiliary MILP problem. We apply a relatively complete recourse property to model (4.2)-(4.4), consisting of adding a dummy arc between the nodes of each OD pair to provide extra capacity at a high premium unit price. The inclusion of these arcs in all recourse problems $\omega \in \Omega$ defines an outsourcing strategy enabling any arbitrary quantity of a commodity to be flowed directly between its associated origin and destination nodes. Thus, $\Phi(y;\omega)$ is always feasible.

The optimality cuts using dummy arcs may involve big-M coefficients. To avoid the insertion of such weak cuts and maintaining the global convergence, we make use of combinatorial cuts. We first identify the infeasibility of a solution using property 4.1.

Property 4.1. *A given solution $\bar{y}$ is infeasible to the MCFNDSD, if there exists at least one SP such that a positive amount of flow is routed on a dummy arc.*

When the current solution is identified as infeasible according to Property 4.1, the optimality cuts for those SPs that involve big coefficients are not added to the MP. Instead, the following combinatorial cut is used :

$$\sum_{a \in \mathcal{A} | \bar{y}_a = 1} (1 - y_a) + \sum_{a \in \mathcal{A} | \bar{y}_a = 0} y_a \geq 1. \tag{4.34}$$

By adding an inequality (4.34) each time the MP solution is infeasible, one ensures that the original feasibility requirements defined in the problem are enforced. Inequality (4.34) can be further strengthened according to the following proposition.

Proposition 4.4. *For an infeasible integer $\bar{y}$, inequality (4.35) is globally valid.*

$$\sum_{a \in \mathcal{A} | \bar{y}_a = 0} y_a \geq 1. \tag{4.35}$$

Proof. Let $\mathcal{A}_0$ and $\mathcal{A}_{01}$ indicate the set of variables fixed to 0 and the set of free variables at the current integer node of the tree. We can easily conclude that the infeasibility is due to not opening enough capacity, since otherwise the dummy arcs would not have been used. This corresponds to opening at least one arc in the set of closed arcs, i.e., $a \in \mathcal{A}_0 \cup (\mathcal{A}_{01} | \bar{y}_a = 0)$, to break the infeasibility, i.e., $\sum_{a \in \mathcal{A}_0} y_a + \sum_{a \in \mathcal{A}_{01} | \bar{y}_a = 0} y_a = \sum_{a \in \mathcal{A} | \bar{y}_a = 0} y_a \geq 1$ because $\bar{y}_a = 0, \forall a \in \mathcal{A}_0$ and $\mathcal{A}_0 \cap \mathcal{A}_{01} = \emptyset$. □

4.4.3 Warm-start strategy

The BD method is known for suffering from ineffective initial iterations due to the generation of low-quality solutions and zig-zagging behavior. To overcome these drawbacks, we propose a *warm-start* (*WS*) strategy to generate an initial set of tight cuts. Unlike the heuristic-base WS strategies, the underlying idea of our approach is to deflect the current master solution. Given an initial starting point y^{ws}, the current master solution $\bar{y}$, and a convex combination weight $0 < \lambda < 1$, we deflect the current solution according to $y^{ws} = \lambda \bar{y} + (1 - \lambda) y^{ws}$. Then,

y^{ws} is used to generate the cuts instead of $\bar{y}$. If the starting solution also satisfies the core point properties, we do not need to solve the auxiliary SP (4.19-4.21) to generate pareto-optimal cuts, since the deflected point guarantees the generation of a non-dominated cut (see Papadakos (2008), Theorem 6). The UB generated using the deflected solution is also valid for the LP relaxation of the problem. Therefore, as long as we are applying this strategy, no auxiliary SP is required to be solved. This yields appreciable savings in computational time. The current solution might be infeasible and thus render the new y^{ws} infeasible as well. In this case, we solve an auxiliary problem to reset y^{ws} at a feasible solution at the vicinity of $\bar{y}$. This auxiliary problem is presented in Appendix B.

This strategy is also capable of considerably alleviating the instability issue of the MP. It dampens the initial large steps of the algorithm through taking an average with a centered solution. Thus, the y^{ws} and the whole procedure can also be interpreted as a stabilizing point and a stabilization strategy (Fischetti et al., 2016).

The strategy is sensitive to the initial value of y^{ws}. We thus propose to solve a MCNF problem with maximum demand, i.e., $d_k = max_{\omega \in \Omega}\{d_k(\omega)\}, \forall k \in \mathcal{K}$. Given the solution $\tilde{y}$ of this problem, we set $y_a^{ws} = y_a^{max} := max\{0.5, \tilde{y}_a\}, \forall a \in \mathcal{A}$.

4.4.4 Managing the branch-and-bound tree

The search tree is one of the most important parts of the algorithm, since its size directly determines its efficiency. To the best of our knowledge, there is no in-depth study in the literature to address the branch-and-bound tree in the cutting plane implementation of the BD method. We examined various cut generation strategies. The preliminary experiments indicated that generating as many cuts as possible at the root node (i.e., first phase of the algorithm) and then generating cuts merely for the potential incumbent solutions yields the best performance. Generating cuts for every or the first 100 fractional solutions inside the tree caused overly slow performance of the algorithm. The reason was due to the low impact of the cuts, generating too many of them and spending too much time on their extraction and handling.

Branching and node selection rules

Node and variable selection decisions are of crucial importance in our algorithm. With respect to the branching, we perform an irregular branching at the root node while, for the rest of the algorithm, we follow the default settings of the optimization solver. To create the first two child nodes, we first identify two sets of variables, $\tilde{\mathcal{A}}_0$ and $\tilde{\mathcal{A}}_1$, which include the variables

that very likely take value 0 or 1 in an optimal solution. Identifying the variables in each set relies on the information gathered during the first phase of the algorithm. Variables which consistently took values smaller than ϵ_{br} or larger than $1-\epsilon_{br}$ are respectively added to $\tilde{\mathcal{A}}_0$ and $\tilde{\mathcal{A}}_1$, such that, for the current solution $\bar{y}$ they satisfy $\sum_{a\in\tilde{A}_1}(1-\bar{y}_a)+\sum_{a\in\tilde{A}_0}\bar{y}_a \leq 1-\epsilon_{br}$, with ϵ_{br} a small positive number. The left branch is then created by adding two constraints of the form $\sum_{a\in\tilde{\mathcal{A}}_0} y_a = 0$ and $\sum_{a\in\tilde{\mathcal{A}}_1} y_a = |\tilde{\mathcal{A}}_1|$, which is equivalent to fixing the variables. The complement to this node indicates that at least one variable in the joint set $\tilde{\mathcal{A}}_0 \cup \tilde{\mathcal{A}}_1$ should change its value. Therefore, the right branch is created by adding a cut of the form $\sum_{a\in\tilde{\mathcal{A}}_1}(1-y_a)+\sum_{a\in\tilde{\mathcal{A}}_0} y_a \geq 1$. The left branch has a small sub-domain, which helps to quickly find good feasible solutions. The right branch has an improved bound since it violates the root solution. As for the node selection strategy, we tested several strategies. At the end, we decided to follow the default of the optimization solver which implements the state-of-the-art strategies.

Upper bounding heuristic

Our branch-and-cut algorithm is dependent on the quality of the incumbent solution to avoid searching inferior regions of the solution space. We thus propose a simple heuristic to generate tight UBs on the optimal integer solution at the end of first phase. The proposed heuristic, unlike the ones in the literature, depends on the BD method itself rather than the problem. Algorithm 1 presents the pseudo-code of this heuristic.

Algorithm 1 : The upper bounding heuristic

Initialize $\hat{\mathcal{A}}_0$, $\hat{\mathcal{A}}_1$, $\hat{\mathcal{A}}_{01}$, ι, T_h
RMP $\leftarrow$ restricted the MP by imposing $\sum_{a\in\hat{\mathcal{A}}_0} y_a = 0$ and $\sum_{a\in\hat{\mathcal{A}}_1} y_a = |\hat{\mathcal{A}}_1|$
for all $t = 0 : \iota$ **do**
  Solve RMP for the given time limit T_h
  if RMP infeasible **then**
    Add strong combinatorial cut $\sum_{a\in\hat{\mathcal{A}}_0} y_a \geq 1$ and *break* the loop
  end if
  Evaluate current solution, generate cuts, and add them to the formulation
  Update the current best UB (if possible)
  if the solution was infeasible for at least one SP based on Property 4.1 **then**
    Add strong combinatorial cut $\sum_{a\in\hat{\mathcal{A}}_0} y_a + \sum_{a\in\hat{\mathcal{A}}_{01}|\bar{y}_a=0} y_a \geq 1$
  else
    Add combinatorial cut $\sum_{a\in\mathcal{A}|\bar{y}_a=1}(1-y_a) + \sum_{a\in\mathcal{A}|\bar{y}_a=0} y_a \geq 1$
  end if
end for
Remove $\sum_{a\in\hat{\mathcal{A}}_0} y_a = 0$ and $\sum_{a\in\hat{\mathcal{A}}_1} y_a = |\hat{\mathcal{A}}_1|$ and go to the branching phase.

We observed that the LP solution is fairly tight and close to an optimal integer solution. Thus, a simple *fix-and-optimize* procedure can yield high quality solutions. The variables that take values smaller than 0.1 or greater than 0.9 are respectively added to the sets $\hat{\mathcal{A}}_0$ and $\hat{\mathcal{A}}_1$. The rest of the variables are added to the set of free variables $\hat{\mathcal{A}}_{01}$. To derive the *Restricted Master Problem* (RMP), variables in $\hat{\mathcal{A}}_0$ and $\hat{\mathcal{A}}_1$ are fixed to 0 and 1, respectively. The RMP is then solved. If feasible, the produced solution is evaluated to get an UB and a set of optimality cuts. To avoid revisiting that solution, a valid combinatorial inequality is also added to the RMP. If the RMP became infeasible, the heuristic procedure terminates and a strong combinatorial cut is added to the formulation. The algorithm iterates for a maximum ι iterations. A time limit of T_h is imposed on each iteration of the RMP. Finally, the imposed bounds on the variables are removed. But, all the generated optimality and combinatorial cuts will remain in the formulation, since they are both useful and valid.

Reduction procedures

The reduction ideas are mainly based on the well-known reduced-cost methodology. However, we do not discuss them here to avoid burdening the article, see, e.g., Gendron et al. (2016). Moreover, one flow conservation constraint in formulation (4.2)–(4.4) is redundant for each commodity k. Thus, we can fix the dual variable associated to its origin (or destination) to zero, i.e., $\pi^k_{O(k)} = 0$, $\forall k \in \mathcal{K}$.

Many of the proposed VIs provide no further convergence advantage after a few iterations. Likewise, a small subset of the Benders cuts are required for global convergence. Thus, a *cleanup* strategy is required to keep the scale of the MP manageable by removing the unnecessary cuts and VIs. The proposed cleanup strategy keeps a record of the slack values for each cut and inequality. If a cut or a VI exhibits a large slack value over some predefined number of iterations, it will be considered as a candidate for removal from the master formulation. In order to avoid frequent cut removal, which profoundly disturbs the optimization solver, the candidate cuts will be removed just before the second phase of the algorithm. Note that this strategy is not applied during the second phase because the cuts are appended to the MP as lazy constraints and the solver eliminates them if they become slack. In addition, the cleanup is applied to the SPs as soon the LP phase is terminated to remove the additional VIs which will be inactive for integer y solutions.

4.5 Numerical results and analysis

In this section, we present the numerical assessment of the proposed algorithm. We first test different versions of the BD algorithm to evaluate the impact of the proposed enhancements on convergence. The second part of the analysis is dedicated to a comparison between our exact algorithm and state-of-the-art algorithms and optimization solvers. We test the robustness and limitations of our method on larger instances in the third part of the experiments.

To carry out the numerical tests, we have used standard benchmark instances, referred to as **R** and **S** in the literature (Crainic et al., 2011, 2014a,b). There are 16 instances in the **S** data set. These instances are defined on fully connected graphs of 16 or 30 nodes, with 14, 40 or 80 commodities, and 10, 20, 60 or 90 scenarios. For presentation purposes, we divided them into three classes based on the number of commodities, which determines the instance difficulty. Eleven instance classes of **R** family, r04 to r14 inclusively, along with five different cost/capacity ratios (1, 3, 5, 7, and 9) were selected. Three numbers of scenarios (16, 32, and 64) are considered for each instance of the **R** set. To generate these scenarios, demands were assumed to be linearly correlated and five levels of positive correlation (0%, 20%, 40%, 60%, and 80%) were used. In summary, 825 different instances of the **R** set are considered.

All programs were coded in C++, using callable libraries of the CPLEX version 12.6.1 as the optimization solver. The code was compiled with g++ 4.8.1, executed on Intel Xeon E7-8837 CPUs running at 2.67GHz with 16GB RAM under a Linux operating system, in single-thread mode.

4.5.1 Implementation details

A number of algorithmic components need to be specified for a complete description of the implementations used to perform the numerical tests. First, the VIs are added to the MP and SPs during the first phase of the algorithm only. LBF and NCI inequalities are added to the master formulation a priori. The other inequalities are added to the MP or SP formulation in a cutting plane method. At the MP level, our pilot experiments indicated that superior performance can be attained through prioritizing the VIs over the Benders cuts, mainly because their extraction is computationally less expensive and because they significantly reduce number of the major iterations. This means that we keep solving the MP and generating the VIs as long as a violated one can be found. This considerably reduces number of the SPs to solve, which constitutes the major computational bottleneck of the algorithm. At the SP level, solving several SPs to generate a single cut is computationally costly. Moreover, in the presence of the SIs, few violated FP inequalities can be found. For

this reason, the SIs are first added to the SP formulations and the FPs are appended in a cutting plane fashion. The cutting plane is executed on each SP only for one iteration per cut generation cycle.

The **S** instances involve arcs that start and end at the same node. These arcs can be removed from the network, when the associated node is not both origin and destination for a specific commodity $k \in \mathcal{K}$. This follows from the non-negative fixed cost values. Moreover, when several commodities share the same OD pair, they can be aggregated into a single commodity if their flow costs are identical.

To specify the stopping criteria, a run time limit of 2 hours, maximum number of 1000 iterations, and optimality tolerance of 1% were considered. The time limit and optimality tolerance for the first phase were fixed to 1 hour and 0.3%. To set the search parameters, extensive computational tests on 5 chosen instances from the **R** family were conducted and the following parameters have been fixed throughout the numerical tests : $\epsilon_{gap} = 10^{-2}$, $\eta = 10^{-2}$, $y^0 = 0.5$, $\lambda = 0.5$, $\epsilon_{br} = 10^{-1}$, $T_h = 5\%$"of total time", $\iota = 5$, $\epsilon_h = 10^{-3}$, $T_I =$ "total iterations of the first phase $- 2$".

The following notation is used in Tables 4.1–4.8 : "Time (Sec.)" gives the CPU time in seconds, "Gap (%)" presents the optimality gap in percentage, "Sol. (%)" indicates percentage of the instances solved to optimality and "Ave." represents the weighted average values. Given the UB and LB values, the optimality gap is calculated as 100(UB - LB)/UB. For the sake of ease in the comparisons, when an algorithm fails to find any feasible solution, the reported optimality gap is set to 100%.

4.5.2 Analysis of the algorithmic enhancements

The goal of this analysis is to assess the effectiveness of the proposed algorithmic enhancements. We investigate to what degree our strategies are further enhancing the state-for-the-art BD method which involves the two-phase approach, multi-cut reformulation, PDS, Pareto-optimal cuts, a heuristic, single search tree and VIs (Crainic et al. (2016)). Therefore, we activated the proposed strategies one by one and observed their cumulative additive value with respect to the convergence of the state-of-the-art BD method for the problem at hand.

For the numerical analyses of this section, we have chosen a subset of the **R** and **S** instances : r04 to r10 with 64 scenarios and s01 and s02. This subset covers all the instances addressed in the literature (i.e., **R** set) and gives the opportunity to assess the behavior of the algorithm on very different network structures (i.e., **S** set).

Feasibility cuts

We first study the proposed strategy to avoid the generation of classical feasibility cuts. As the numerical results in Table 4.1 indicate, adding the relatively complete recourse to the model and using strong combinatorial cuts outperforms the use of classical feasibility cuts. This can partially be explained by the fact that most of the master solutions are slightly infeasible, meaning that the constructed network at the MP fails to cover only a small portion of the demand. In this case, the generated optimality cuts seem to improve the lower bound faster than the inclusion of feasibility cuts. Moreover, the current solutions to the MP that are infeasible are usually so for only a small subset of the SPs. Thus, following the proposed strategy, optimality cuts are still being added for all feasible SPs. In turn, this accelerates the rate at which the relaxation defined by the MP can be improved.

Table 4.1 Assessing the performance of the strategy to avoid generating feasibility cuts

	Benders with feasibility cuts			Benders with complete recourse		
	Time (s)	Gap (%)	Sol. (%)	Time (s)	Gap (%)	Sol. (%)
r04	160.43	0.65	100.00	86.01	0.65	100.00
r05	224.16	0.79	100.00	190.57	0.76	100.00
r06	3135.42	1.52	76.00	2655.89	1.39	72.00
r07	993.86	0.91	92.00	618.58	0.85	100.00
r08	2877.81	1.86	64.00	2494.79	1.41	76.00
r09	5989.05	1.98	24.00	5602.56	1.70	36.00
r10	5879.67	3.72	20.00	5694.71	3.84	24.00
s01	56.03	0.17	100.00	56.86	0.15	100.00
s02	7200.00	54.59	0.00	7200.00	54.40	0.00
Ave. **R.**	2751.49	1.63	68.00	2477.59	1.51	72.57
Ave. **S.**	4342.41	32.82	40.00	4342.74	32.70	40.00

Pareto-optimal cuts

Table 4.2 displays the numerical comparison between our cut generation strategy denoted "Proposed approach (I)", and that of Magnanti and Wong (1981). We also examined a variant of our approach in which the core point is dynamically updated according to $y^0 \leftarrow \frac{1}{2}(\bar{y} + y^0)$, denoted as "Proposed approach (II)".

Two key observations can be made from Table 4.2. First, the proposed cut generation scheme outperforms the original method of Magnanti and Wong (1981). For 125 out of 175 **R** instances that both methods solve, our approach reduced the average CPU time by 21.23%. For the other 50 instances, it improved the average optimality gap and CPU time by 2.32% and 9.53%, respectively. The superiority of our approach is particularly evident for the instances

Table 4.2 Comparing cut generation schemes

	Magnanti & Wong (1981)			Proposed approach (I)			Proposed approach (II)		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	86.01	0.65	100.00	68.50	0.63	100.00	77.76	0.65	100.00
r05	190.57	0.76	100.00	136.90	0.72	100.00	158.21	0.69	100.00
r06	2655.89	1.39	72.00	2355.46	1.24	76.00	2391.80	1.26	76.00
r07	618.58	0.85	100.00	511.80	0.80	100.00	431.30	0.81	100.00
r08	2494.79	1.41	76.00	2278.20	1.36	76.00	2386.64	1.34	76.00
r09	5602.56	1.70	36.00	5376.49	1.69	36.00	5248.34	1.69	44.00
r10	5694.71	3.84	24.00	5643.28	3.38	32.00	5501.45	3.23	36.00
s01	56.86	0.15	100.00	53.07	0.12	100.00	10.68	0.03	100.00
s02	7200.00	54.40	0.00	7200.00	52.44	0.00	7200.00	51.17	0.00
Ave. **R** :	2477.59	1.51	72.57	2338.66	1.41	74.29	2313.64	1.38	76.00
Ave. **S** :	4342.74	32.70	40.00	4341.23	31.51	40.00	4324.27	30.72	40.00

for which the algorithm executes more iterations involving the dummy arcs. In this case, the resolution of the auxiliary SP (4.19)-(4.21) is considerably slower due to the equality constraint (4.21). Second, updating the core point yields a positive impact on the overall performance, requiring 2.55% less iterations, on average, to converge. This is probably due to the fact that the updated core point gives a better measure to emulate the optimal direction at the current iteration. Note that, the core point is updated if the current solution is feasible according to Property 4.1.

The same observations can be made for the **S** instances. The three methods seem to have the same average running time. This is mainly due to the instances in the s02 class. They are larger in scale than those in s01, and consume the whole running time. The Magnanti and Wong Magnanti and Wong (1981)'s approach fails to find a feasible solution for 3 instances in s02, while this value for the proposed approaches reduces to 2.

We also tested the methods in Papadakos (2008) and Sherali and Lunday (2013) to generate Pareto-optimal cuts (results available from the first author), and observed that the former is less efficient. This is due to the generation of non-exact cuts with respect to the current solution, particularly when the algorithm approaches a local optima. Sherali and Lunday (2013)'s method underperfomed that of Magnanti and Wong (1981), indicating the cuts generated in the latter method are stronger. Our method alleviates all these shortcomings.

Warm start

The WS strategy was proposed to overcome the ineffective initial iterations. Thus, we first illustrate its impact for the first phase of the algorithm, as it is deactivated after 15 iterations or when this phase ends. The initial point in this experiment is set equal to the core point, i.e., $y^{ws} = y^0$. The comparative results are depicted in Figure 4.1. On average, the algorithm

with the WS requires 53.32% less time to optimize the first phase for the **R** instances and 43.40% less for the **S** instances. More significant savings are obtained on the larger instances. It is worth mentioning that the infeasibility rate of the algorithm during this phase dropped by 75.08% and 83.10% for the **R** and **S** instances, respectively.

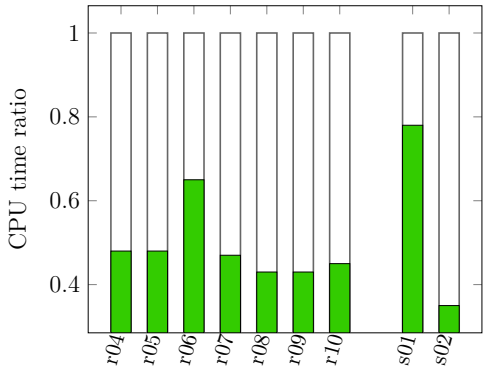

Figure 4.1 CPU time to solve the first phase with warm-start activated (dark bars) or not (light bars)

We next examine the impact of the WS strategy on the convergence of the algorithm. Two variants of the algorithm are compared : one in which the initial point is set to the core point ($y^{ws} = y^0$), and a second one in which it is set to the point proposed in section 4.4.3 ($y^{ws} = y^{max}$). The numerical results are given in Table 4.3.

Table 4.3 Impact of the warm-start on the convergence given initial point setting

	WS ($y^{ws} = y^0$)			WS ($y^{ws} = y^{max}$)		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	63.68	0.64	100.00	72.34	0.62	100.00
r05	94.39	0.70	100.00	95.60	0.58	100.00
r06	2121.76	1.14	80.00	2124.45	1.10	80.00
r07	421.61	0.77	100.00	404.51	0.78	100.00
r08	2392.24	1.24	76.00	2395.59	1.19	76.00
r09	5195.91	1.42	44.00	4992.15	1.32	52.00
r10	5355.16	2.81	36.00	5443.17	2.85	40.00
s01	39.27	0.30	100.00	15.57	0.05	100.00
s02	7200.00	38.00	0.00	7200.00	8.12	0.00
Ave. **R** :	2234.96	1.24	76.57	2218.26	1.21	78.29
Ave. **S** :	4335.71	22.91	40.00	4326.23	4.89	40.00

Comparing the results in Tables 4.3 and 4.2 , we can confirm that the WS strategy has a positive impact on the performance of the BD method. The largest gap reduction was observed for the **S** class, an average improvement of 25.74% for the s02 family. The overall time improvement may seem somehow limited in light of the results in Figure 4.1. This

can be explained by two facts. The time spent on the first phase compared to the overall time requirement is rather small, particularly for the **R** instances. Moreover, many of the previously unsolved instances still remain unsolved, although with lower gaps. This does not favorably affect the average time.

When the WS strategy is initialized with y^{max}, more of the **R** instances are solved. For the **S** instances, a further optimality gap improvement of 78.63% is observed in the s02 category. The reason we observe more significant changes for the **S** instances can be explained by their fully connected network structure. For such instances, the time spent on the SPs is noticeably larger when the solution is infeasible. Since using the WS significantly reduces the number of infeasible iterations, the LP phase is optimized faster : 69.97% on average. In addition to having a tighter LP relaxation, more time remains for the second phase of the algorithm to find better incumbent solutions. Last remark, the average time requirement for some instances, e.g., r10, has slightly increased. This indicates that although the proposed initialization performs the best on average, it could be further improved, the algorithm being sensitive to the initial point. Thus, further research on this subject would be worthwhile.

Valid inequalities

Next, we examine the impact of the VIs on the performance of the algorithm. We experimented with three versions of the algorithm. Only the VIs at the SP level are activated in the first version, the VIs are only added to the MP in the second and, the VIs are applied to the both MP and SPs in the third. The numerical results are summarized in Table 4.4.

Table 4.4 Impact of the VIs on the algorithm performance

	VIs for the SPs			VIs for the MP			VIs for both SPs and MP		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	60.02	0.57	100.00	41.11	0.58	100.00	29.26	0.53	100.00
r05	110.34	0.39	100.00	53.34	0.57	100.00	34.40	0.38	100.00
r06	2765.27	1.29	64.00	1876.71	0.91	80.00	2687.94	0.93	80.00
r07	460.53	0.55	100.00	326.85	0.71	100.00	210.30	0.53	100.00
r08	1337.47	0.70	92.00	1220.91	0.89	96.00	551.66	0.58	100.00
r09	3174.41	1.72	64.00	4233.37	1.17	68.00	2542.66	1.49	76.00
r10	5196.56	2.40	44.00	4229.81	1.52	68.00	3967.93	1.56	64.00
s01	9.99	0.02	100.00	16.52	0.04	100.00	16.33	0.02	100.00
s02	285.93	0.11	100.00	7200.00	6.35	0.00	189.40	0.13	100.00
Ave. **R** :	1872.08	1.09	80.57	1711.73	0.91	87.43	1432.02	0.86	88.57
Ave. **S** :	175.55	0.08	100.00	4326.61	3.83	40.00	120.17	0.09	100.00

The VIs add significant value to the algorithm. In all cases, the CPU time and average optimality gaps were reduced, while the percentage of solved instances increased. The best results are attained when VIs are added at both MP and SP levels.

VIs at the MP and SP level display different behaviors for the **R** and **S** instances. In the former case, the VIs at the SP level have less impact than those at the MP level, while the opposite appears true for the **S** instances. To explain these observations, Figure 4.2 compares the versions 1 (VIs added to the SPs) and 2 (added to the MP) displaying on the left the relative improvement (in %) of the LP relaxation at the root node, and on the right the relative increase (in %) CPU times.

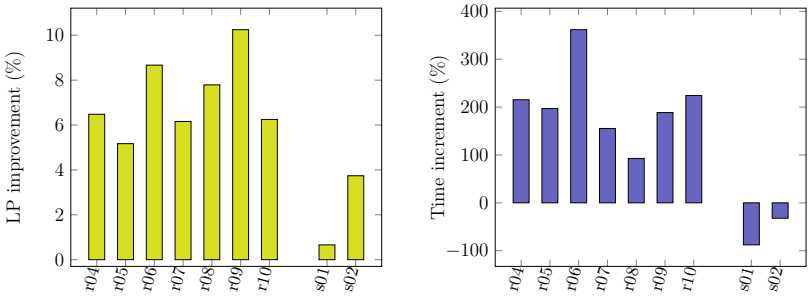

Figure 4.2 Comparing LP relaxations (left) and CPU times (right) when VIs are added to SP or MP

Figure 4.2 illustrates that adding VIs to the SPs is more effective, by more than 7.25% (on average), in tightening the LP relaxation, than adding to the MP. This, however, increases the CPU time to optimize the LP phase for the **R** instances by 204.97%. The LP improvement compared to the optimal solution is small, while the increased time requirement is very large. For the **S** family, both time and LP relaxation have improved, because the addition of the stronger optimality cuts reduces the number of iterations to optimize the LP phase by an average of 72.69%. Also, the cutting plane at the SP level has been so effective that the LB at the root node is less than 1.92% and 0% from the final LB for the **R** and **S** instances, respectively. The same values for the cutting plane at the MP level are 8.75% and 0.75%. When both cutting plane methods are activated, the LP relaxation is further tightened and the CPU time remains reasonably low. This has thus given the best results.

Note that adding VIs to the SPs deteriorates the performance over r06 instances despite of significantly tightening the LP relaxation. We realized that this anomaly is due to the default search mechanism of CPLEX. If we turn off the default cuts and heuristics, adding VIs to the SPs speeds up the algorithm by nearly a factor of 2. We thus call for more researches to study the exploration strategies for the branch-and-bound tree in context of the BD method.

In summary, examining the **S** and **R** instances, we conclude that the VIs are more effective in dense networks. The NCIs are effective only when the number of nodes is larger than the number of commodities. Moreover, the decomposition provides the capability to efficiently handle an exponential number of the well-known family of VIs, such as SIs. The LBF cuts

can significantly improve the initial LB. Figure 4.3 depicts the quality of this bound during the initial iterations of the algorithm when these cuts are appended to the MP versus the case when they are ignored. To conduct this experiment, all proposed enhancements are turned off and the algorithm is run for one iteration only. On average, the initial LB is 92.44% higher than the case where the LBFs are not added to the MP. The average time spent on the generation of these inequalities is also less than 6.10 seconds.

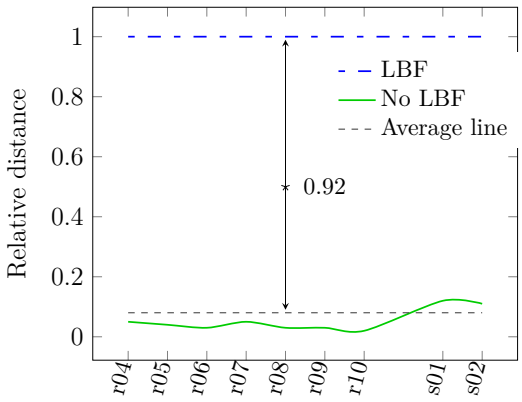

Figure 4.3 Improvement of the initial lower bound when the LBFs are appended

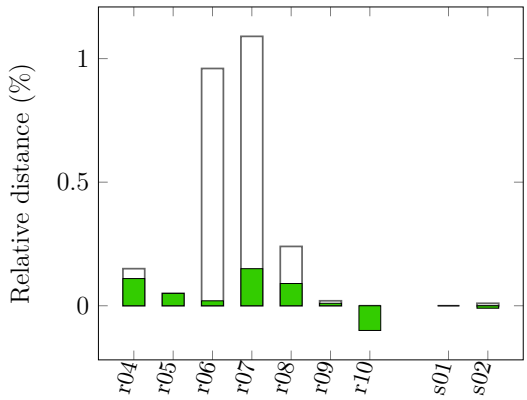

Figure 4.4 Distance of the UB obtained by Heuristic (I) (white bars) and (II) (dark bars) from the best known value

Heuristic

Multiple integer solutions can be extracted at each iteration of the heuristic. They are valid to generate both cuts and UBs. We have thus tested two versions of the heuristic, using only the best solution in Heuristic (I), while Heuristic (II) uses all the extracted solutions. Figure 4.4 depicts the relative distance (in %) of the UBs attained by each version compared to the best known value.

Heuristics (I) and (II) are able to find solutions that are respectively 0.36% and 0.05% from the best known UB. Heuristic (I) finds better or equal UBs for 110 instances out of 175. This value goes up to 124 for Heuristic (II), which is, however, more time consuming by 36.29% (on average) than Heuristic (I). For the **S** instances, both consistently find the best known incumbents, and Heuristic (II) for two instances finds even better bounds. We next examine the impact of the heuristics on the performance of the algorithm. The results for both versions are summarized in Table 4.5.

The algorithm with Heuristic (I) improves the average run time, while it solves a larger portion of the instances with Heuristic (II). The major advantage of the heuristics is that

Table 4.5 The impact of the heuristic on algorithm performance

	Heuristic (I)			Heuristic (II)		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	28.77	0.59	100.00	37.12	0.59	100.00
r05	34.85	0.46	100.00	46.79	0.47	100.00
r06	2474.42	1.02	80.00	2633.02	0.98	80.00
r07	174.29	0.64	100.00	218.41	0.63	100.00
r08	490.69	0.64	100.00	791.41	0.58	100.00
r09	2390.35	1.41	76.00	2373.02	1.40	80.00
r10	3990.64	1.42	64.00	4064.08	1.41	64.00
s01	41.47	0.02	100.00	17.22	0.02	100.00
s02	201.00	0.20	100.00	242.76	0.18	100.00
Ave. **R** :	1369.14	0.88	88.57	1451.98	0.87	89.14
Ave. **S** :	137.18	0.13	100.00	152.54	0.12	100.00

they remove many candidate solutions and a massive part of the branch-and-bound tree. If we compare the number of explored nodes with and without the heuristic, we observe an average reduction of 23.12% when the heuristic is activated. However, the heuristic does not always yield a net computational advantage, mainly because of its non-trivial running time. For example, consider **r10** for which using Heuristic (II) yields a better UB than the best solution previously found without the heuristic. This did not positively affect the number of solved instances, although the average optimality gap is reduced. The time spent on the heuristic accounts for 20.47% of the total running time, on average. For some instances this value goes up to 50%. Considering the time spent on the first phase, limited time remains for the second phase which is not enough to close the optimality gap of challenging instances. When we increase the time limit to 5 hours, we observe that the algorithm with Heuristic (II) solves a wider range of instances and entails lower CPU time.

Value of the PDS

The initial motivation for the PDS was alleviating the weak relaxation of the MP. In this paper, this drawback has also been tackled by means of VIs and a WS strategy. In order to verify whether or not it is still beneficial to use the PDS in our algorithm, we have turned off the PDS feature and ran our algorithm once again. The results are presented in Table 4.6.

We observe that for small and medium instances, the PDS tends to increase the cost of the iterations. This is clearly due to the additional complexity it inserts into the MP. This observation relates to the instances with very tight or very loose capacity ratio. However, if the cut generation cycle is much more time consuming than the MP, the PDS usually improves the performance. This is certainly true for the **S** instances since the cut generation

Table 4.6 Impact of the PDS on convergence

	Without PDS			With PDS		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	25.95	0.59	100.00	37.12	0.59	100.00
r05	48.43	0.51	100.00	46.79	0.47	100.00
r06	2451.07	1.01	76.00	2633.02	0.98	80.00
r07	218.84	0.68	100.00	218.41	0.63	100.00
r08	393.53	0.64	100.00	791.41	0.58	100.00
r09	2197.37	1.71	80.00	2373.02	1.40	80.00
r10	4326.32	1.84	56.00	4064.08	1.41	64.00
s01	18.63	0.14	100.00	17.22	0.02	100.00
s02	1759.21	0.20	100.00	242.76	0.18	100.00
Ave. **R** :	1380.21	0.99	87.43	1451.98	0.87	89.14
Ave. **S** :	1062.98	0.17	100.00	152.54	0.12	100.00

cycle is the most time consuming part of the algorithm. Nonetheless, the algorithm with the PDS solves a wider range of instances. The PDS thus appears a valid and efficient acceleration technique.

Marginal impact of the acceleration strategies

To complete the computational experiments of this section, we study the marginal impact of each strategy on the algorithm's running time if we turn off that specific acceleration. We observed that the average running time increases by factors of 1.31, 2.49. 7.16, 8.89, 2.68, 2.18 and 1.71 if we deactivate the heuristic, VIs for the MP, VIs for the SP, VIs for both MP and SP, WS, PDS and our Pareto-optimal cut generation strategy, respectively. Thus, all the accelerations have a positive marginal impact on the performance. The Pareto-optimal cut generation strategy and the heuristic have the least marginal impact, while VIs have the greatest impact on accelerating the BD method, particularly the VIs for SPs.

4.5.3 Comparison with other approaches

We assess the performance of our algorithm versus alternate exact approaches, i.e., BD method without the acceleration techniques, CPLEX and the algorithm of Crainic *et al.* Crainic et al. (2016). The best strategy for each enhancement, as introduced in the previous section, is included in our algorithm. In order to examine the efficiency and robustness of the algorithms, a wider range of instances was considered : 525 **R** (r04–r10) and 12 **S** (s01-s02) instances. The results are summarized in Table 4.7, where "TR" and "GR" indicate the time and gap ratios calculated by dividing the results of each approach by those of our algorithm.

The column labeled "SD (%)" gives the number of solved instances by our approach minus the alternate approaches in %. Consequently, the bigger the value for these measures, the higher the efficiency of our algorithm.

Table 4.7 Comparing the proposed BD algorithm to alternate exact methods

	Naive BD			**Cplex 12.6**			**Crainic et al. [2016]**		
	TR	GR	SD (%)	TR	GR	SD (%)	TR	GR	SD (%)
r04	68.55	4.35	0.00	6.17	1.10	0.00	3.78	1.61	0.00
r05	303.17	53.58	60.00	18.74	1.01	0.00	4.17	6.01	0.00
r06	63.14	18.96	48.00	10.56	3.68	0.00	6.00	2.89	18.67
r07	301.97	3.86	8.00	3.94	1.43	0.00	4.98	2.99	2.67
r08	370.71	170.38	84.00	21.80	2.19	20.00	18.88	9.22	13.33
r09	59.55	14.49	48.00	8.11	3.98	10.67	20.11	1.43	26.67
r10	37.64	369.73	64.00	2.92	8.48	25.33	2.55	5.39	38.67
s01	395.87	252.73	50.00	8.03	3.96	0.00	7.11	8.51	0.00
s02	61.29	284.22	66.67	10.30	6.15	100.00	88.52	298.46	50.00
Ave. **R**.	172.10	90.76	44.57	10.32	3.12	8.00	8.64	4.22	14.29
Ave. **S**.	228.58	268.47	58.33	9.39	5.28	60.00	55.95	182.48	30.00

The figures displayed in Table 4.7 illustrate that the proposed algorithm outperforms other methods for all criteria. The naive BD method is the less efficient. Compared to the method proposed in Crainic et al. (2016), our algorithm is 2.55 to 88.52 times faster. Compared to the CPLEX, these values range from 2.92 to 21.80. The average optimality gaps of our approach are also 1.01 to 8.48 and 1.43 to 298.46 times lower than those of CPLEX and Crainic et al. (2016), respectively. For **R** (**S**) instances, the maximum optimality gap for CPLEX, Crainic et al. (2016) and our algorithm are respectively 57.56% (11.87%), 8.88% (100%) and 4.51% (0.49%). Our algorithm is then able to solve more instances, e.g., 48 and 78 challenging instances in comparison to CPLEX and Crainic et al. (2016), which cannot find feasible solutions for half of the s02 instances.

The convergence behavior of the exact algorithms over time is illustrated in Table 4.5 for instance 49 from class r10 (stopping criteria were fixed at 10 hours run time and optimality gap of 0.03%). We see that, while CPLEX starts off with tighter bounds than Crainic et al. (2016), it terminates with the largest optimality gap. Our algorithm quickly finds tight bounds. The initial bounds are much tighter than alternate approaches and it quickly converges to an optimal solution. It is interesting to observe that the alternate methods also find the optimal solution, although at much slower pace. However, they cannot prove its optimality due to slow progression of the LB.

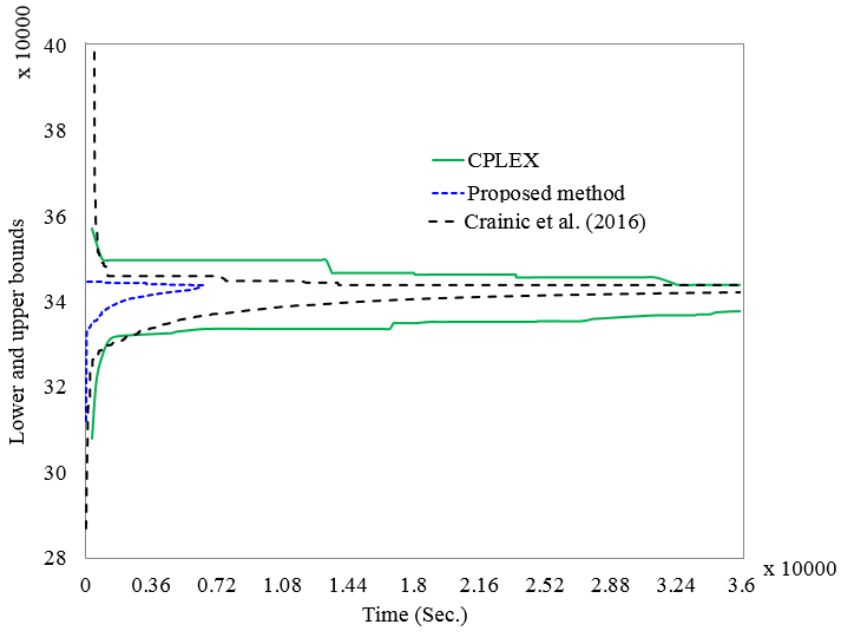

Figure 4.5 Convergence behavior of the exact algorithms over time

Large instances

We conclude the experimental study by focusing on larger instances. We present the results for a time limit of 10 hours, for the largest instance classes in our sets, r11 - r14 and s03. For a more comprehensive view of the topic, we also included the instances that remained unsolved in previous sections. The numerical results for our BD method, CPLEX and Crainic et al. (2016) are summarized in Table 4.8.

Table 4.8 Computational experiments on larger instances

	Proposed BD algorithm			CPLEX 12.6			Crainic et al. (2016)		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r06	2366.40	0.67	100.00	9436.76	0.57	85.33	3494.55	0.80	98.67
r09	4888.16	0.86	94.67	13983.34	1.64	78.67	6238.04	0.98	97.33
r10	10074.77	0.92	88.00	24408.12	4.65	45.33	16021.18	1.35	70.67
r11	14155.99	1.96	62.67	20576.50	11.24	60.00	17436.35	2.86	54.67
r12	13844.30	2.31	74.67	21792.24	11.95	52.00	21124.33	3.80	46.67
r13	28814.03	4.09	20.00	29440.17	19.29	20.00	28155.89	5.38	28.00
r14	28907.46	5.63	21.33	34487.90	31.54	12.00	29487.41	10.45	20.00
s03	4958.44	0.69	100.00	29682.71	5.51	33.33	36000	54.211915	0.00

We observe that our algorithm performs better than other methods. The average optimality gap for the **R** instances for the three methods are 2.35%, 11.55% and 3.66%, respectively, while the average time requirements are 14721.59, 22017.86 and 17422.54 CPU seconds. Our algorithm is capable of solving 65.90% of the instances, more than the 50.48% of CPLEX

and 59.43% by the method of Crainic et al. (2016). It is noticeable that our method is the only one able to solve all the s03 instances, and to do it in short times, which emphasizes its robustness in handling different instances.

Looking closely to the convergence behavior of the proposed algorithm on large instances, we observed difficulties for some instances, mainly due to the a less-well-performing second phase inducing long plateaus without any improvement in the bounds. We examined, in particular, the 187 **R** instances (out of 525) for which the algorithm failed to reach optimality. We observed, on average for these unsolved instances, an LB value at the root node of 1108493.01 at time 2043.91, and a final lower bound of 1116109.30 at time 36099.04. The very low improvement rate corresponding to these figures indicates the need for more research into improving the second phase of the proposed BD algorithm.

4.6 Conclusions and remarks

We presented various acceleration strategies to boost the convergence of the Benders decomposition method. The strategies include those already certified in the literature for their positive impact on the algorithm as well as new techniques. We added two cutting-plane methods to the Benders framework to overcome three main drawbacks which make its application less effective than alternative approaches, i.e., weak linear relaxation of the problem at hand, weak optimality cuts, and weak relaxation of the master problem. To overcome the ineffective initial iterations, a warm-start strategy was used, which allows generating a set of tight cuts quickly. We also proposed to generate Pareto-optimal cuts through fixing variables in the auxiliary problem of Magnanti and Wong (1981) by exploiting the dual information of the primary SP. A simple heuristic was also developed to find high-quality incumbent solutions. To avoid extraction and inclusion of feasibility cuts, we developed relatively complete recourse property for the problem, and proposed strengthened alternate cuts to the classical feasibility ones to avoid generating optimality cuts with big coefficients.

The proposed algorithm was successfully tested on a wide range of stochastic network design instances. However, we believe our theoretical/numerical results are general in nature and will remain valid independently of the problem chosen. We observed that the proposed algorithm is at least 10 times faster than the state-of-the-art algorithms and commercial solvers. We also realized that valid inequalities at both master and subproblem level are very important. The results indicated that VIs for the SP are even more important when tight ones are present. Moreover, we noticed that carefully updating the core point at each iteration can improve the performance of the BD algorithm through better selection of the cuts.

There are several directions to be explored to further enhance the proposed algorithm. We observed that the algorithm is sensitive to the core point in terms of both initialization and updating. Despite the importance of this subject, there is still no theoretical guiding on how to initiate and update this point. The second phase of the algorithm takes the largest portion of the running time. Although it starts with a fairly tight UB, it hardly improves this bound. The lower bound also progresses very slowly. Thus, developing advanced acceleration strategies specialized for the second phase of the algorithm appears an important subject for future research. Another line of research revolves around improving the proposed heuristic so as to make it more rapid while keeping the same level of accuracy. In the same line of research, studies on selection criteria for carefully choosing solutions from a pool appear promising, in order to avoid using many useless solutions to generate bounds and cuts. With respect to the feasibility cuts, there are other proposals in the literature which would be worthwhile to examine and compare to the strategy proposed in this article, see e.g., Ruszczyński (1997). Furthermore, we call for experimental studies to compare the proposed warm-start/stabilization method to other stabilization techniques. In the same spirit, it is worthwhile to conduct a numerical comparison of the proposed method to the scenario decomposition approach (Ahmed, 2013). Last but no least, we found great opportunities in improving the proposed algorithm by using parallelization techniques.

CHAPTER 5 ARTICLE 3: THE BENDERS DUAL DECOMPOSITION METHOD

Ragheb Rahmaniani[1], Shabbir Ahmed[3], Teodor Gabriel Crainic[2], Michel Gendreau[1], Walter Rei[2]

[1] CIRRELT & Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal H3C 3A7, Canada.

[2] CIRRELT & School of Management, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal H3C 3P8, Canada.

[3] School of Industrial & Systems Engineering, Georgia Institute of Technology, 765 Ferst Drive, Atlanta, GA 30332, United States.

Abstract. Many methods that have been proposed to solve large-scale MILP problems rely on decomposition techniques. These methods exploit either the primal or the dual structure of the problem, yielding the Benders decomposition or Lagrangian dual decomposition methods. We propose a new and high performance approach, called Benders dual decomposition (BDD), which combines the complementary advantages of both methods. The development of BDD is based on a specific reformulation of the Benders subproblem, where local copies of the master variables are introduced in the proposed subproblem formulation and then priced out into the objective function. We show that this method : (i) generates stronger feasibility and optimality cuts compared to the classical Benders method, (ii) can converge to the optimal integer solution at the root node of the Benders master problem, and (iii) is capable of generating high quality incumbent solutions at the early iterations of the algorithm. We report encouraging numerical results on various benchmark MILP problems.

Keywords. *Benders decomposition*, *Lagrangian duality*, *Mixed-integer programming.*

History. Submitted to Operations Research.

5.1 Introduction

Mixed-integer linear programing (MILP) is used to model a wide range of engineering and financial problems (Nemhauser and Wolsey, 1988). Owing to the importance and inherent complexities of MILP models, it has been the subject of intense research since the early 1950s (Beale, 1965; Jünger et al., 2009; Newman and Weiss, 2013). In this article, we consider MILP

problems of the following generic form

$$\min_{y,x}\{f^\top y + c^\top x :\ By \geq b,\ Wx + Ty \geq h,\ y \in \mathbb{Z}^n_+,\ x \in \mathbb{R}^m_+\} \tag{5.1}$$

where $f \in \mathbb{R}^n$, $B \in \mathbb{R}^{k\times n}$, $b \in \mathbb{R}^k$, $c \in \mathbb{R}^m$, $W \in \mathbb{R}^{l\times m}$, $h \in \mathbb{R}^l$, $T \in \mathbb{R}^{l\times n}$. We assume that the above problem is feasible and bounded. A prominent general approach to solve problem (5.1) relies on partitioning techniques such as the Benders decomposition method (Benders, 1962), especially when part of the input data defined in the program are stochastic (Ruszczyński and Świętanowski, 1997; Costa, 2005; Birge and Louveaux, 1997; Rahmaniani et al., 2017a). To solve problem (5.1) with the Benders decomposition (BD) method , we introduce an auxiliary variable θ and the following master problem (MP)

$$MP = \min_{y,\theta}\{f^\top y + \theta :\ By \geq b,\ \theta \geq \bar{\theta},\ y \in \mathbb{Z}^n_+\} \tag{5.2}$$

where $\bar{\theta}$ is a lower bound on θ to avoid unboundedness of the problem. The MP, the objective value of which defines lower bounds for (5.1), is solved in a branch-and-cut method. At each integer node of the branch-and-bound tree, the solution y^* is fixed in the following dual subproblem (DSP)

$$DSP \ = \ \max_{\alpha}\{(h - Ty^*)^\top \alpha :\ W^\top \alpha \leq c,\ \alpha \in \mathbb{R}^l_+\} \tag{5.3}$$

If the above problem is unbounded, a direction of unboundedness r is chosen and the feasibility cut $(h - Ty)^\top r \leq 0$ is added to the MP to exclude all infeasible y solutions satisfying $(h - Ty)^\top r > 0$. Otherwise, a feasible solution to (5.1) is identified (thus allowing the upper bound of the algorithm to be updated), and the optimality cut $(h - Ty)^\top \bar{\alpha} \leq \theta$ is added to the MP. This procedure is repeated until the algorithm converges to an optimal solution. Due to the impact that the quality of the starting lower bound has on the size of the branch-and-bound tree generated by the algorithm, optimality and feasibility cuts can also be generated at fractional nodes of the tree at the beginning of the solution process to rapidly improve the quality of the lower bound. Such a strategy has mainly been applied at the root node, see (McDaniel and Devine, 1977; Naoum-Sawaya and Elhedhli, 2013; Adulyasak et al., 2015; Gendron et al., 2016; Bodur et al., 2017).

The BD method has been the subject of intense research due to its practical importance in handling various MILP problems, e.g., production routing (Adulyasak et al., 2015), power generation (Cerisola et al., 2009), healthcare (Lin et al., 2016), logistics (Cordeau et al., 2006), green wireless network design (Gendron et al., 2016), map labeling (Codato and Fischetti, 2006), supply chain management (Santoso et al., 2005), investment planning (Mitra

et al., 2016), network design (Costa, 2005), and so on. It may however perform disappointingly without the inclusion of some (problem-specific) acceleration techniques, see the recent literature reviews by Rahmaniani et al. (2017a); Fragkogios and Saharidis (2018) and references therein for a complete information. Generally speaking, as highlighted in these reviews, the poor performance of the BD method is due to its shortfalls from both dual and primal perspectives.

The performance of the BD method, from a dual perspective, depends on the quality of the cuts chosen to bound the projected term (Holmberg (1990); Crainic et al. (2016); Fischetti et al. (2016)). In particular, if the underlying LP relaxation of the problem is weak and/or the subproblems are degenerated, the algorithm performs poorly because the cuts and the root node bound are very weak (Magnanti et al., 1986; Van Roy, 1983; Sahinidis and Grossmann, 1991; Geoffrion and Graves, 1974; Cordeau et al., 2006; Rahmaniani et al., 2017b). Thus, effective selection of the Benders cuts has been the main focus of several studies, e.g., Magnanti et al. (1986), Wentges (1996), Codato and Fischetti (2006), Papadakos (2009), Fischetti et al. (2010), Contreras et al. (2011), Sherali and Lunday (2013), among others. In more recent studies, strengthening the Benders cuts has been performed by making use of valid inequalities (VIs), see e.g., Bodur et al. (2017) and Rahmaniani et al. (2017b). Lagrangian techniques have also been used to generate alternative optimality cuts, particularly when the subproblem includes integrality requirements (Cerisola et al., 2009; Zou et al., 2016). In fact, it has been shown that the cuts obtained from the Lagrangian dual subproblems are not only valid for the Benders master problem, they are also generally tighter than the classical ones (Van Roy, 1983; Santoso et al., 2005; Mitra et al., 2016).

From a primal point of view, the BD method has no systematic mechanism to generate high quality upper bounds. Indeed, it has oftentimes been observed that the evolution of the upper bound throughout the BD search process stagnates and finding good quality solutions can be quite a challenge, see (Boland et al., 2015). Thus, problem-specific heuristics have been used to generate a pool of high-quality solutions or to improve the quality of the master solutions obtained, see e.g. Poojari and Beasley (2009); Rei et al. (2009); Costa et al. (2012); Pacqueau et al. (2012) and section 6.3 in Rahmaniani et al. (2017a).

Motivated by the important role that the cuts, the root node bound and the incumbent solution play on the performance of the BD method, we propose a new and high performance decomposition strategy, referred to as Benders dual decomposition (BDD), to overcome the aforementioned primal and dual shortfalls. The development of BDD is based on a specific reformulation of the subproblems where local copies of the master variables are introduced. This reformulation of the subproblems has been used in previous studies to generate gene-

ralized Benders cuts (Geoffrion, 1972; Flippo and Rinnooy Kan, 1993; Grothey et al., 1999; Zaourar and Malick, 2014; Fischetti et al., 2016; Zou et al., 2016). In the present case, we apply Lagrangian duality to the proposed subproblem reformulation to price out the coupling constraints that link the local copies to the master variables. This allows us to impose the integrality requirements on the copied variables to obtain MILP subproblems, which are comparable to those defined in *Lagrangian dual decomposition* (LDD) (Ruszczyński, 1997; Rush and Collins, 2012; Ahmed, 2013). As a consequence of obtaining these MILP subproblems, we will show that our proposed strategy efficiently mitigates the primal and dual inefficiencies of the BD method. Also, in contrast to the LDD method, BDD does not require an enumeration scheme (e.g., branch-and-bound) to close the duality gap. Furthermore, our strategy enables a faster convergence for the overall solution process. In summary, the main contributions of this article are the following :

- proposing a family of strengthened optimality and feasibility cuts that dominate the classical Benders cuts at fractional points of the MP,
- showing that the proposed feasibility and optimality cuts can give the convex hull representation of the MP at the root node, i.e., no branching effort being required,
- producing high quality incumbent values while extracting the optimality cuts,
- developing numerically efficient implementation methodologies for the proposed decomposition strategy and presenting encouraging results on a wide range of hard combinatorial optimization problems.

The remainder of this article is organized as follows. In Section 5.2 we present the proposed decomposition strategy and in Section 5.3 we present a toy example to illustrate its benefits. In Section 5.4, the developed implementation methodologies are presented and discussed. The experimental design and computational results are, respectively, provided in Sections 5.5 and 5.6. The last Section includes our conclusions and a discussion of future research directions.

5.2 The proposed BDD method

The primal Benders subproblem, i.e., the primal form of (5.3), is $\min_x\{c^\top x :\ Wx \geq h - Ty^*,\ x \in \mathbb{R}^m_+\}$. By defining y^* as the current MP solution, without loss of generality, it can be rewritten as

$$\min_{x,z}\{c^\top x :\ Bz \geq b,\ Wx + Tz \geq h,\ z = y^*,\ x \in \mathbb{R}^m_+, z \in \mathbb{R}^n_+\}. \tag{5.4}$$

It should be noted that the constraints $Bz \geq b$ are redundant due to the presence of $z = y^*$. For a given feasible solution y^*, by solving problem (5.4), the following optimality cut can

be derived

$$\theta \geq c^\top \bar{x} + (y - \bar{z})^\top \lambda^* \tag{5.5}$$

where $\bar{x}$ and $\bar{z}$, represent the optimal solution of the subproblem (5.4) and λ^* are the dual multipliers associated to the constraints $z = y^*$. If problem (5.4) for y^* is infeasible, then the following feasibility subproblem needs to be solved

$$\min_{x,z,v}\{\mathbb{1}^\top v : \ Bz \geq b, \ Wx + Tz + v \geq h, \ z = y^*, \ x \in \mathbb{R}^m_+, z \in \mathbb{R}^n_+, \ v \in \mathbb{R}^l_+\} \tag{5.6}$$

to generate a feasibility cut of the form

$$0 \geq \mathbb{1}^\top \bar{v} + (y - \bar{z})^\top \beta^* \tag{5.7}$$

where $\bar{v}$ define the optimal values of the v variables, β^* are the dual multipliers associated to the constraints $z = y^*$ and $\mathbb{1}$ is a vector of ones of size l. The optimality and feasibility cuts (5.5) and (5.7) are often referred to as *generalized Benders cuts* (GBC) (Geoffrion, 1972; Sahinidis and Grossmann, 1991; Grothey et al., 1999; Fischetti et al., 2016). It should be noted that the dual multipliers associated to the equality constraints $z = y^*$ define a subgradient of the objective function.

5.2.1 Strengthening the optimality and feasibility cuts

To strengthen the optimality cut (5.5), we price out the constraints $z = y^*$ into the objective function using the dual multipliers λ. By doing so, the following Lagrangian dual problem of (5.4) is obtained

$$\max_{\lambda \in \mathbb{R}^n} \min_{x \in \mathbb{R}^m_+, z \in \mathbb{R}^n_+} \{c^\top x - \lambda^\top (z - y^*) : \ Bz \geq b, \ Wx + Tz \geq h\} \tag{5.8}$$

In the interest of brevity, we henceforth use $X := \{(x, z) \in \mathbb{R}^m_+ \times \mathbb{R}^n_+ \mid Bz \geq b, \ Wx + Tz \geq h\}$ and $F := \left\{y \in \mathbb{R}^n_+ | \{By \geq b\} \bigcap \{Wx \geq h - Ty : \text{for some } x \in \mathbb{R}^n_+\}\right\}$ to represent the compact form of the feasible region of subproblem (5.4) and the set of all feasible solutions y to the LP relaxation of problem (5.1). By applying this relaxation step, integrality requirements can be imposed on any subset of the variables z, given that they are no longer set equal to y^* (i.e., a master solution that is not guaranteed to be feasible).

In the following proposition, it is shown that problem (5.8) with $z \in \mathbb{Z}^n_+$ can be used to produce a valid optimality cut :

Proposition 5.1. *For any solution $y^* \in F$, let λ^*, $\bar{x}$ and $\bar{z}$ be optimal solutions obtained by*

solving the following max-min MILP problem

$$\max_{\lambda \in \mathbb{R}^n} \min_{(x,z)\in X} \{c^\top x - \lambda^\top (z - y^*) : z \in \mathbb{Z}^n_+\}, \tag{5.9}$$

then

$$\theta \geq c^\top \bar{x} + (y - \bar{z})^\top \lambda^* \tag{5.10}$$

is a valid optimality cut for the Benders master problem.

Proof. See Appendix D. □

We observe from the previous proof that, when considering an integer solution $y^* \in F \bigcap \mathbb{Z}^n_+$, the optimality cut (5.10) is at most as strong as the classical Benders optimality cut. However, when applied using a fractional solution to the master problem, the cut (5.10) provides an added advantage which is studied in the following Theorem. This is an important observation to make, given that the LP relaxation of the MP is often solved first to quickly obtain valid cuts which enable the Benders method to perform more efficiently (McDaniel and Devine, 1977; Naoum-Sawaya and Elhedhli, 2013). The next result reports the improvement that is obtained regarding the lower bound provided through the use of the optimality cuts (5.10) when compared to the lower bound provided by (5.5).

Theorem 5.1. *Given the dual multiplier $\hat{\lambda} \in \mathbb{R}^n$ obtained from problem (5.4), the optimality cut (5.10) is parallel to optimality cut (5.5) and at least $\sigma \geq 0$ units tighter, where*

$$\sigma = \min_{(x,z)\in X:\ z\in\mathbb{Z}^n_+} \{c^\top x - \hat{\lambda}^\top z\} \quad - \quad \min_{(x,z)\in X} \{c^\top x - \hat{\lambda}^\top z\} \tag{5.11}$$

Proof. See Appendix E. □

As a direct implication of Theorem 5.1, the optimality cuts (5.10) are at least as strong than (5.5). If we use the dual multipliers obtained from solving subproblem (5.4), then according to Theorem 5.1 we can lift the optimality cut by an amount equal to the duality gap of the inner minimization problem. We refer to such cuts as *strengthened Benders cut*. If we optimize the Lagrangian problem (5.9) to generate an optimality cut, then we refer to such cut as *Lagrangian cut* (Zou et al., 2016).

The same strategy can also be used to strengthen the Benders feasibility cuts (5.7). In this

case, the following Lagrangian dual for problem (5.6) is used to evaluate $z = y^*$

$$\max_{\beta \in \mathbb{R}^n} \min_{(x,z,v) \in \mathbb{R}^m_+ \times \mathbb{R}^n_+ \times \mathbb{R}^l_+} \{\mathbb{1}^\top v - \beta^\top (z - y^*) : \ Bz \geq b, \ Wx + Tz + v \geq h\} \tag{5.12}$$

Following Proposition 5.1, it might appear natural to impose integrality requirements on the z variables. However, given that the constraint set $Bz \geq b$ may not be satisfied, this approach does not guarantee that a valid feasibility cut will be obtained. Therefore, in the following proposition, a modified Lagrangian dual problem is proposed to generate a valid and lifted feasibility cut for the MP.

Proposition 5.2. *For an arbitrary $y^* \notin F$, let β^*, $\bar{v}$, $\bar{u}$, $\bar{x}$ and $\bar{z}$ be the optimal solution of*

$$\max_{\beta \in \mathbb{R}^n} \min_{(x,z,v,u) \in \mathbb{R}^m_+ \times \mathbb{Z}^n_+ \times \mathbb{R}^l_+ \times \mathbb{R}^k_+} \{\mathbb{1}^\top v + \mathbb{1}^\top u - \beta^\top (z - y^*) : \ Bz + u \geq b, \ Wx + Tz + v \geq h\} \tag{5.13}$$

then,

$$0 \geq \mathbb{1}^\top \bar{v} + \mathbb{1}^\top \bar{u} + (y - \bar{z})^\top \beta^* \tag{5.14}$$

is a violated valid feasibility cut to the master problem.

Proof. See Appendix F. □

From the last part in the proof of Proposition 5.2, it is clear that the proposed feasibility cut dominates the classical one if the duality gap of the minimization problem is non-zero. Thus, the results of Theorem 5.1 directly apply to this case as well.

To conclude this section, we study in Theorem 5.2 the root node bound of the BD method when the proposed optimality and feasibility cuts are used. To ease the exposition of the result, we henceforth use $H := \{(x, z, v, u) \in \mathbb{R}^m_+ \times \mathbb{R}^n_+ \times \mathbb{R}^l_+ \times \mathbb{R}^k_+ |\ Bz + u \geq b, \ Wx + Tz + v \geq h\}$ to represent the feasible region of problem (5.13) and $Y := \{y \in \mathbb{R}^n_+ |\ By \geq b\}$ to indicate the domain of the relaxed y variables.

Theorem 5.2. *Let Z_{IP} and Z^{LP}_{BD} be, respectively, the optimal objective value of problem (5.1) and the root node bound of the Benders MP with the proposed Lagrangian optimality and feasibility cuts, then $Z_{IP} = Z^{LP}_{BD}$.*

Proof. See Appendix G. □

From the proof of Theorem 5.2, one observes that our method is closely related to the LDD method (see Appendix J for more details on this method). This decomposition approach

has been successfully applied in the context of solving stochastic programs (Ahmed, 2013), where it is referred to as *scenario decomposition*. As shown in Ahmed (2013), Lagrangian dual decomposition enables a stochastic program to be separated via the scenarios, allowing lower bounds to be computed more efficiently within a general search process. In the present case, as demonstrated in Theorem 5.2, when applied in the context of the Benders algorithm, this decomposition strategy strengthens the cuts generated while solving the LP in such a way as to close the gap at the root node. While solving (5.9) and (5.13) each time to optimality to generate the associated cuts may not be computationally efficient, the previous theoretical results nevertheless provide clear guidelines defining how Benders cuts can be lifted to improve the quality of the lower bound generated at each iteration of the algorithm. Furthermore, it should be noted that the proposed cut generation strategy is applicable to a wider range of problems. Specifically, from the definitions of problems (5.9) and (5.13), it is clear that the generation of optimality and feasibility cuts is independent of the specific structure of set X. Thus, set X could include nonlinear constraints and integer requirements on the variables.

5.3 Example

To illustrate the benefits of the proposed method, consider following toy problem :

$$\min_{y\in\{0,1\},\ x\geq 0}\{x : x+15y \geq 8, 3x+10y \geq 13, x+10y \geq 7, 2x-10y \geq -1, 2x-70y \geq -49\} \quad (5.15)$$

The optimal solution to this problem is $y = 0$, which has a cost of 8 units. We solve the LP relaxation of this problem using the (i) classical, (ii) strengthened and (iii) Lagrangian cuts. The detailed results of each iterative procedure can be found in Appendix K. In Figure 5.1, we have graphically depicted the cuts generated at each iteration of these methods in the (y, θ)-space.

In solving the LP relaxation of the master problem with the classical Benders cuts, after 5 iterations the method converges to the optimal LP solution $y = 0.58$ and its objective value of 2.4 units. If the strengthened Benders cuts are used, then the first two iterations of the algorithm generate the same cuts as the classical ones. The reason for this is that the y solutions for these two iterations are integer and as a result the Benders cuts are the strongest. At the third iteration, solving the MP produces the solution $y = 0.65$. For this solution the associated strengthened Benders cut is parallel to the classical Benders cuts but tighter by 6.125 units. At this point, the proposed method with the strengthened cuts converges. For the master solution $y = 0.65$, the Lagrangian cut provides the convex hull representation

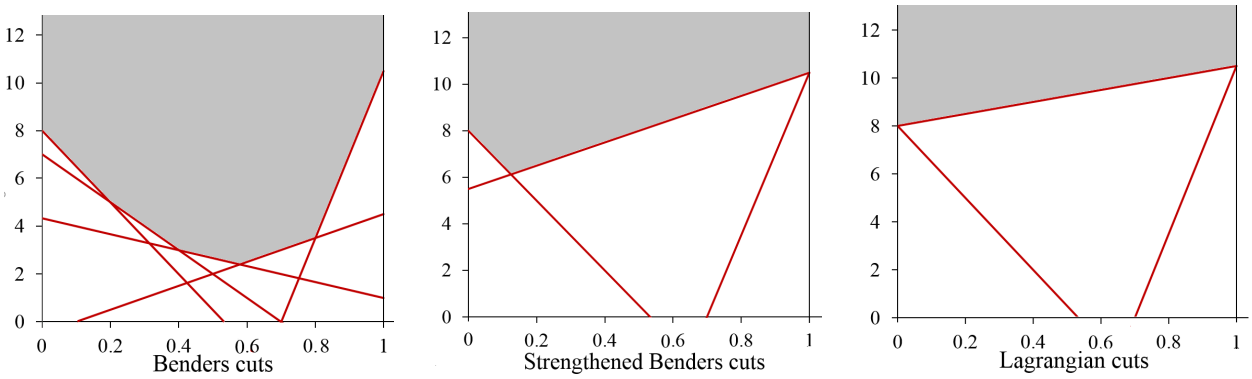

Figure 5.1 Performance of different cuts in cutting the solution space of LP master problem

of the MP and, consequently, the algorithm converges to the optimal integer solution. As demonstrated in the previous section, in this case, while solving the LP relaxation of the MP, the optimal integer solution to the problem (5.15) is obtained.

5.4 Implementation details

As clearly shown in section 5.2 and numerically illustrated on a simple example in section 5.3, the proposed method reduces the number of required cuts for convergence and significantly tightens the root node bound. However, this is achieved at the cost of solving one (or several) MILP subproblem(s), which may create a numerical bottleneck for the Benders algorithm (especially when solving large-scale optimization problems). Therefore, in this section, we develop a series of strategies that aim to apply the proposed method in a computationally efficient manner.

5.4.1 Multi phase implementation

To alleviate the computational burden of optimizing the Lagrangian dual problem (5.9) for numerous iterations, we propose to implement a three-phase strategy to generate the Benders cuts. This is motivated by the fact that, at the initial iterations of the Benders algorithm, the master solutions are usually of very low quality. At this point, the derived cuts provide a poor approximation of the value function at the optimal region of the problem. In turn, it may not be worthwhile to invest the effort of lifting these cuts. However, as the Benders algorithm progresses and more and more cuts are added to the MP (thus improving the quality of the lower bound provided by the MP and guiding the search process towards more promising areas of the feasible region), then a lifting strategy may be applied to accelerate the convergence of the algorithm. The proposed multi phase implementation works as follows.

Phase 1. To quickly derive valid cuts, we first solve the LP relaxation of the MP with the classical cuts (5.5) and (5.7), which is a strategy that was originally proposed by McDaniel and Devine (1977). Given that this strategy has become one of the main methods used to efficiently apply the Benders algorithm on numerous applications, see (Rahmaniani et al., 2017a), we thus apply it in this phase of our implementation.

Phase 2. We then generate the strengthened optimality Benders cuts by first fixing the dual multipliers λ to the optimal values obtained by solving the problem (5.4). Using the obtained λ values, the inner minimization problem in (5.9) (i.e., the Lagrangean dual subproblem) is then solved to find the values of $\bar{x}$ and $\bar{z}$, which are then used to generate a strengthened optimality Benders cuts. Similarly, if the MP solution is infeasible at this point, a strengthened feasibility cut is generated by first fixing the dual multipliers β to the optimal values obtained by solving the problem (5.6). Using values β, the inner minimization problem in (5.13) is then solved to find the values of $\bar{v}$, $\bar{u}$ and $\bar{z}$, which are then applied to produce a strengthened feasibility cut. Overall, this second phase starts the lifting process of the proposed cut generation strategy.

Phase 3. In this last phase, Lagrangian cuts are generated. To do so, we heuristically solve the Lagrangian dual problem (5.9). Therefore, to generate an optimality cut and assuming that a series of values $\bar{x}^v$ and $\bar{z}^v$, for $v = 1, 2, \ldots, t-1$, have been obtained via the process by which a strengthened cut is generated (i.e., solving the inner minimization problem (5.9) for a series of values λ^v, for $v = 1, 2, \ldots, t-1$), the following regularized problem is solved to update the dual multipliers λ :

$$\max_{\lambda \in \mathbb{R}^n, \eta \in \mathbb{R}^1} \ \eta - \frac{\delta}{2}||\lambda^{t-1} - \lambda||_2^2 \tag{5.16}$$

$$\eta \leq c^\top \bar{x}^v + (y - \bar{z}^v)^\top \lambda \qquad \forall v = 1, 2, \ldots, t-1. \tag{5.17}$$

The objective function (5.16) seeks to maximize the value by which the cut is lifted (i.e., value η) minus the distance value $\frac{\delta}{2}||\lambda^{t-1} - \lambda||_2^2$. The latter component of the objective function, which defines a trust-region for the Lagrangean multipliers, is included to stabilize the updating process for a given prefixed value $\delta \in \mathbb{R}_+^1$ that represents the stabilization parameter. It thus enables a new vector λ^t of values to be found that is close to the one obtained in the previous iteration (i.e., λ^{t-1}). As for the constraints (5.17), they provide an inner approximation of the Lagrangian dual subproblem. Given the new Lagrangian multiplier values λ^t, the inner minimization problem in (5.9) is instantiated and then solved to obtain $\bar{x}^t$ and $\bar{z}^t$. At this point, the constraint $\eta \leq c^\top \bar{x}^t + (y - \bar{z}^t)^\top \lambda$ is added to (5.17) and the updated regularized program is solved. This process is repeated until either the new multiplier values

fail to lift the cut by at least $\gamma\%$ when compared to the previous iteration, or, a maximum number of iterations (i.e., defined as parameter κ) is reached. Finally, it should be noted that the same procedure is applied to generate Lagrangian feasibility cuts by simply interchanging the appropriate programs.

5.4.2 Partially relaxed subproblems

To further reduce the computational burden of solving one or several MILP subproblems to generate a single cut, we suggest in this section two relaxation strategies that can be applied. The first strategy applies the integrality requirements only on a subset of the z variables following the relaxation of the constraint set $z = y^*$. We thus partition the variables as follows : $z^\top = [z_I, z_J]^\top$ using two sets I and J such that $I \subseteq \{1, \ldots, n\}$, $J \subseteq \{1, \ldots, n\}$, $I \cap J = \emptyset$ and $I \cup J = \{1, \ldots, n\}$, where z_I and z_J represent the subvectors of z that include the variables whose indexes are included in the subsets I and J, respectively. The integrality requirements are then imposed solely on the variables of the subvector z_I. Therefore, the formulation of the cut generation problem becomes : $\max_{\lambda \in \mathbb{R}^n} \left\{ \lambda^\top y^* + \min_{(x,z)\in X} \{c^\top x - \lambda_I^\top z_I - \lambda_J^\top z_J : z_I \in \mathbb{Z}_+^{|I|}\} \right\}$. Given that the integrality constraints on the variables z_J are relaxed, this problem provides an optimality cut that is weaker than the Lagrangian cut. However, it remains stronger when compared to the Benders cut. At each iteration of the Benders algorithm, to define the subsets I and J we apply the following procedure : we initially set $I = \emptyset$ and $J = \{1, \ldots, n\}$; then if a master variable y_a (where $a \in \{1, \ldots, n\}$ is the index associated with a variable in the y vector) takes a fractional value such that $\epsilon_0 + \lfloor y_a^* \rfloor \leq y_a^* \leq \lceil y_a^* \rceil - \epsilon_1$, where y^* is the current master solution and the non-negative values ϵ_0 and ϵ_1 are two preset parameters such that $\epsilon_0 + \epsilon_1 < 1$, then a is removed from J and added to I. Therefore, parameters ϵ_0 and ϵ_1 are used to adjust the width of the fractional interval that is used to identify the variables on which the integrality restrictions will be imposed.

As a follow-up to the previous approach, a second relaxation strategy is proposed. This strategy, which is problem specific, is based on the idea of fixing variables in the Lagrangian subproblems. To apply it, we need to assume that fixing a z variable to its upper limit in the inner minimization problem of (5.9) widens the solution space, i.e., $\min_{(x,z)\in X}\{c^\top x : z = y^*\} \geq \min_{(x,z)\in X}\{c^\top x : z \geq y^+\}$, where the elements of y^+ are equal or strictly greater than those in y^*. Such an assumption can be made in a wide gamut of applications, particularly in cases where capacity constraints are imposed. When this assumption is observed, then the following result can be applied :

Proposition 5.3. *Let y^* be the current master solution and $\hat{\lambda}$ be the corresponding dual*

multiplier obtained from linear program (5.4). Furthermore, let I' be a subset of I such that $y_a^* \geq ub_a - \tilde{\epsilon}$ and $\hat{\lambda}_a \geq \upsilon$ for every $a \in I'$ and for given values $\tilde{\epsilon} \geq 0$ and $\upsilon \geq 0$, where ub_a is the upper bound on variable y_a. Then, the following restricted Lagrangian dual problem

$$\max_{\lambda \in \mathbb{R}^n} \Big\{ \min_{(x,z) \in X : z_I \in \mathbb{Z}_+^{|I|}} \{c^\top x - \lambda^\top (z - y^*) : \ z_{I'} = ub_{I'}\} : \lambda_{I'} = \hat{\lambda}_{I'} \Big\}, \tag{5.18}$$

can be used in replacement of (5.9) to generate approximate Lagrangian cuts, where $ub_{I'}$ is the vector of upper bounds associated with the variables whose indexes are in I'.

Proof. See Appendix H □

Given that the Lagrangian dual problem needs to be solved several times per iteration, we expect that these strategies will noticeably enhance the overall numerical performance of the Benders algorithm. However, relaxing the subproblems will weaken the optimality cuts derived. Therefore, careful consideration must be given to the specific values to which the different parameters defining the strategies will be fixed. In the case of the second strategy, it should be noted that the parameters ϵ and υ define the conditions by which the variables whose indexes will be included in subset I' are chosen. Subset I' is then used to instantiate the restricted Lagrangian dual problem (5.18), which is solved to generate the approximate Lagrangian cuts. The more y variable indexes are added to subset I' and the more problem (5.18) is restricted, thus reducing the numerical burden to produce a cut. However, the larger the restriction that is applied and the weaker the cut that is derived. Clearly, a trade-off between numerical efficiency and cut strength needs to be established when applying the strategy on a specific application.

5.4.3 ϵ-optimal cuts

In many applications finding optimal solutions for the Lagrangian dual subproblems, even when applying the aforementioned strategies, may not be easy. In our pilot numerical experiments, we observed that closing the optimality gap from, for example, 1% to 0% is often the most time consuming part of solving a subproblem. Therefore, in this section, we propose to use ϵ-optimal solutions of the Lagrangian dual subproblems to generate approximate cuts. The following results shows the validity of this approach :

Proposition 5.4. *Let $y \in Y$ and $(\bar{x}(\epsilon), \bar{z}(\epsilon)) \in X$ be an ϵ-optimal solution of the minimization problem of the Lagrangian dual. Then, the following cut can be derived :*

$$\theta \geq c^\top \bar{x}(\epsilon) + (y - \bar{z}(\epsilon))^\top \lambda^* - \epsilon \qquad \forall y \in Y \tag{5.19}$$

Proof. See Appendix I. □

5.4.4 Upper bounding

In this last section, we detail how the cut generation strategy can also be used to improve the upper bound that is generated throughout the Benders solution process. We first highlight the fact that solving the inner minimization problem in (5.9) provides an integer $\bar{z}$ solution, which is feasible for the original problem (5.1) given the copied constraints in the proposed reformulation (5.8). Therefore, the cut generation process produces solutions that can be used as incumbents for the overall problem. Thus, the upper bound can be easily updated by evaluating the objective value of these solutions, i.e., $f^\top \bar{z} + c^\top \bar{x}$. If this value is lower than the current incumbent, then the associated solution is used to generate a classical optimality cut as well.

It should be noted that for cases where there are multiple subproblems (i.e., stochastic models), then the solution from one subproblem may be infeasible for the overall problem given that it may not satisfy all the constraints in the other subproblems. Moreover, to obtain the objective value of a solution in this case, all subproblems associated with the solution need to be evaluated. We thus first propose to record the extracted $\bar{z}$ solutions in a pool. From this pool we then merely evaluate those solutions which are generated at least τ times. This is motivated by the heuristic notion that a solution with a higher level of saturation might correspond to an optimal one. Finally, if the objective value of the solution is already higher than the current incumbent value, then it can be simply discarded considering that no improvement is possible.

5.5 Experimental design

To study whether our method is numerically beneficial, we complement the theoretical analysis presented in previous sections with an extensive computational study. In this section, we describe the MILP problems used to conduct our numerical analyses, the general characteristics of the test instances and how the algorithms were implemented.

5.5.1 Problems studied and test instances

We test our method on benchmark instances of three different optimization problems from the scientific literature. Detailed descriptions of these problems can be found in Appendix L. Here, we provide a high level summary of these problems and instances.

Our first test instances are related to the *fixed-charge multicommodity capacitated network design* (FMCND) problem with stochastic demands. This problem naturally appears in many practical applications (Klibi et al., 2010) and it has been numerically shown to be notoriously hard to solve (Crainic et al., 2011). In addition, it lacks the complete recourse property, which entails that the generation of feasibility cuts is necessary to ensure the convergence of the BD method. We have considered 7 classes of instances (r04 to r10) from the **R** set, as developed in Crainic et al. (2001). Each class includes 5 instances with varying cost and capacity ratios.

Our second test instances are related to the *capacitatied facility location* (CFL) problem, which was introduced by Louveaux (1986) and addressed in Bodur et al. (2017); Fischetti et al. (2016) and Boland et al. (2015) among others. To avoid the generation of feasibility cuts, which do not contribute towards the improvement of the lower bound generated by the BD method, the complete recourse property can be enforced via the inclusion of a constraint in the MP. As for the instances, we use the deterministic **CAP** instances (101 to 134), which are part of the OR-Library. These instances include 50 customers with 25 to 50 potential facilities. For the stochastic variant, we have used the scenarios generated by Bodur et al. (2017) where each scenario includes 250 scenarios. It should be noted that the deterministic instances of this problem are referred to as “CFL" and the stochastic ones as “CFL-S".

Finally, our third set of benchmark instances are associated to the *stochastic network interdiction* (SNI) problem proposed by Pan and Morton (2008). It is important to note that this problem is structurally different from the previous ones, in the sense that there are no fixed costs associated to the master variables in the objective function. Moreover, due to the presence of a budget constraint, the variable fixing strategy as detailed in Section 5.4.2 cannot be applied. Regarding the instances, we have considered those which have been described and used by Pan and Morton (2008), Bodur et al. (2017) and Boland et al. (2015). All instances have 456 scenarios and 320 binary master variables associated to the same network of 783 nodes and 2586 arcs. We specifically consider those instances which are part of the classes referred to as “snipno" 3 and 4 (see Pan and Morton (2008) Tables 3 and 4). Each class includes 5 different instances and for each instance we have considered varying budget limits (i.e., 30, 40, 50, 60, 70, 80 and 90 units).

5.5.2 Parameter settings and implementation details

All algorithms (both the BDD and Benders methods) are implemented in a branch-and-cut framework, i.e., a branch-and-bound tree is built and Benders cuts are generated only at the root node and whenever a potential incumbent solution is found. We apply our proposed strategy exclusively at the root node of the branch-and-bound tree. To optimize the Lagran-

gian dual in the LDD method, we use the same technique as the one previously discussed in section 5.4.1.

In order to make the implementations simple and easily replicable, we do not employ any specialized codes or algorithms. Thus, we solve all the derived problems with a general-purpose solver. Accordingly, we also avoid incorporating any acceleration technique in our algorithm in order to perform a fair assessment of our proposed decomposition method versus the classical one.

In all methods, cuts (both feasibility and optimality) are generated by solving each subproblem within an optimality gap of 0.5% (i.e., $\epsilon = 0.5\%$). Moreover, to generate the Lagrangian cuts for the FMCND and CFL instances, we apply the variable fixing strategies defined in section 5.4.2. To provide a thorough numerical assessment of our proposed decomposition method, we have implemented the following four variants of the strategy :

- BDD_1 **:** uses the strengthened Benders cuts by imposing the integrality requirements on all the copied variables in the subproblem, i.e., $J = \emptyset$ and $I = \{1, .., n\}$,
- BDD_2 **:** uses the strengthened Benders cuts by imposing the integrality requirements on a subset of the copied variables in the subproblem according to the strategy detailed in section 5.4.2
- BDD_3 **:** similar to BDD_1 but also generates Lagrangian cuts,
- BDD_4 **:** similar to BDD_2 but also generates Lagrangian cuts.

To set the values for the different search parameters described in section 5.4, we conducted a series of preliminary experiments over a small subset of the instances and we have chosen the following values for the parameters in our algorithms : $\delta = \frac{1}{t+1}10^{-2}$, $\kappa = 10$, $\gamma = 10^{-1}$, $\epsilon_0 = 10^{-2}$, $\epsilon_1 = 10^{-1}$, $\tilde{\epsilon} = 10^{-2}$, $\tau = 3$, $\upsilon = 10^2$, where t is the iteration counter. As for the stopping criteria, an amount of 10 hours was allotted for the overall solution of the considered problems. It should be noted that this amount included a time limit of 1 hour to optimize and lift the root node bound. The overall optimality gap to terminate the algorithms was fixed at 1%.

Finally, all programs were coded in C++, using the CPLEX version 12.7 as the optimization solver. The codes were compiled with g++ 4.8.1 and executed on Intel Xeon E7-8837 CPUs running at 2.67GHz. with 32GB RAM under a Linux operating system and in a single-thread mode. The branch-and-cut algorithm was also implemented using the CPLEX callable libraries.

5.6 Computational results

In this section, we quantify the computational benefits of the proposed decomposition methodology when solving the instances considered. We first analyze the performance and behavior of the various variants of our method (i.e., BDD_1, BDD_2, BDD_3 and BDD_4). To do so, we assess the quality of the lower and upper bounds obtained by our variants at the root node when compared with the classical decomposition methods. We will then evaluate the convergence behavior of our approach and compare its performance with the state-of-the-art optimization solver CPLEX.

5.6.1 Computational results at the root node

In Table 5.1, both the time requirements and the percentages of the root gap with respect to the optimal solutions are reported for the BD, LDD and the four variants of our method when these strategies are used to solve all considered benchmark instances. Recall that these experiments were obtained by allotting a maximum of 1 hour of running time.

Table 5.1 Average percentage gap of the lower bound at the root node from the optimal solution in different methods

Prob.	Inst.	Benders decomposition		Dual decomposition		The proposed BDD method							
						BDD_1		BDD_2		BDD_3		BDD_4	
		Gap(%)	Time(s.)	Gap(%)	Time(s.)	Gap(%)	Time(s.)	Gap(%)	Time(s.)	Gap(%)	Time(s.)	Gap(%)	Time(s.)
FMCND	r04	22.48	6.27	5.89	2479.79	19.52	49.46	19.89	43.95	1.64	900.98	2.25	773.49
	r05	17.95	46.68	9.82	2859.31	13.17	239.90	13.78	232.09	1.14	2160.62	1.26	2105.21
	r06	20.96	336.02	10.73	3349.31	14.63	1026.42	14.86	960.73	9.72	2977.36	8.86	2965.59
	r07	18.56	16.19	5.39	3146.00	16.50	70.10	16.82	61.48	3.58	1048.47	4.62	922.98
	r08	20.57	87.14	9.38	3067.86	17.35	376.39	17.44	331.62	7.12	2673.75	6.22	2051.87
	r09	23.43	596.56	12.85	3876.46	16.99	1683.00	16.93	1832.72	12.81	3681.55	11.80	3571.77
	r10	(45.22)	(3600.56)	(19.37)	(3759.72)	-	-	-	-	-	-	-	-
	Mean	**20.66**	**181.48**	**9.01**	**3129.79**	**16.36**	**574.21**	**16.62**	**577.10**	**6.00**	**2240.46**	**5.83**	**2065.15**
CFL-S	101-104	22.63	16.31	5.17	3588.80	21.38	63.49	21.44	53.15	1.03	696.72	0.76	475.54
	111-114	8.45	77.79	5.74	3806.09	7.73	310.84	7.73	283.79	1.05	1582.18	0.97	1955.86
	121-124	19.23	77.69	12.50	3650.62	18.38	276.00	18.36	231.65	1.51	3003.90	1.38	2762.77
	131-134	24.14	69.67	17.77	3674.22	23.76	172.27	23.75	173.34	2.32	2229.10	1.83	3342.80
	Mean	**18.61**	**60.37**	**10.30**	**3679.93**	**17.81**	**205.65**	**17.82**	**185.48**	**1.47**	**1877.97**	**1.23**	**2134.24**
CFL	101-104	23.80	0.44	0.00	0.03	23.10	0.98	23.49	0.83	2.89	16.48	9.80	14.73
	111-114	11.22	3.11	0.35	0.16	10.57	6.15	10.55	4.83	2.20	174.14	2.20	226.24
	121-124	21.29	2.35	0.00	0.07	21.29	1.18	21.07	1.57	3.41	146.26	2.99	154.51
	131-134	24.36	1.29	0.00	0.04	24.36	0.83	24.15	1.24	4.31	112.69	6.12	199.56
	Mean	**20.17**	**1.80**	**0.09**	**0.08**	**19.83**	**2.28**	**19.82**	**2.12**	**3.20**	**112.39**	**5.28**	**148.76**
SNI	3-30	22.47	128.20	19.59	3751.47	22.42	162.82	22.42	149.65	14.37	1065.85	14.35	1221.82
	3-40	26.21	143.18	23.10	3837.34	26.21	182.12	26.21	170.39	17.35	1362.20	17.35	1377.63
	3-50	27.53	155.60	23.46	3827.19	27.53	193.94	27.53	177.32	19.20	1303.32	19.30	1367.68
	3-60	28.17	164.83	24.42	3890.09	28.17	204.55	28.17	177.51	22.14	1095.66	22.18	1100.10
	3-70	28.88	167.15	25.30	3877.71	28.88	204.70	28.88	173.34	21.90	979.65	22.15	964.42
	3-80	30.87	179.76	26.95	3786.87	30.85	212.44	30.85	178.77	20.53	1316.73	20.81	1294.92
	3-90	33.13	198.12	28.66	3841.09	33.13	226.75	33.13	173.33	23.95	1056.19	24.24	1100.43
	4-30	25.15	82.10	21.80	3815.90	25.14	130.79	25.14	129.57	14.49	1089.79	14.88	1047.74
	4-40	28.45	90.05	26.99	4015.53	28.45	144.97	28.45	143.55	18.47	1254.47	18.46	1274.26
	4-50	30.54	98.57	27.47	3730.99	30.54	161.58	30.54	147.27	21.29	1195.25	22.11	1054.68
	4-60	32.07	97.85	30.44	3790.82	32.07	160.11	32.07	152.50	25.30	1007.63	25.06	979.38
	4-70	32.61	102.49	32.50	3808.80	32.61	158.54	32.61	142.50	25.24	830.67	24.94	919.71
	4-80	33.27	105.02	33.03	3846.78	33.27	169.71	33.27	143.75	22.76	916.80	22.50	916.77
	4-90	36.17	108.51	35.90	3832.19	36.16	163.54	36.16	133.21	22.77	1082.82	21.53	1268.45
	Mean	**29.68**	**130.10**	**27.12**	**3832.34**	**29.67**	**176.90**	**29.67**	**156.62**	**20.70**	**1111.22**	**20.70**	**1134.86**

We observe that BDD_1 and BDD_2 tighten the root node bound by more than 4% for FMCND and less than 1% for the CFL-S and CFL instances. However, they fail to improve this bound for the SNI instances. This observation can be explained by the fact that, in SNI the duality gap of the subproblems is very small. Thus, according to Theorem 5.1, the strengthened optimality cuts are close to the classical Benders cuts.

This being said, we observe that the Lagrangian cuts are very effective in tightening the root node bound. The most significant improvements are attained for the CFL-S instances, where the average gap at the root node is reduced to less than 1.5% from 17.81% by using the Lagrangian cuts (see the results of BDD_3 and BDD_4 methods). The same type of significant improvements are also observed for the CFL and FMCND instances. Even in the case of the SNI instances, where the obtained improvements are less significant, the root node bound is still tightened by approximately 9%. It should be noted that these stronger root node bounds are achieved at the cost of higher running times. For example, on average, the time requirement of BDD_3 is more than 30 times higher than the BD method for the CFL-S instances. We thus investigate in section 5.6.2 if this additional effort at the root node pays off in the overall performance of the algorithm.

Our method, when applied using the Lagrangian cuts, also outperforms the LDD algorithm considering that it generates much tighter root node bounds in shorter times. These results can be explained by the fact that our approach effectively integrates the complementary advantages of both the BD and LDD methods, i.e., an effective search mechanism coupled with the generation of strong cuts. Regarding these results, it should be noted that the LDD method solves the deterministic variants of the facility location problem (i.e., CFL) very quickly. This is explained by the fact that this specific decomposition strategy when applied on deterministic problems is equivalent to solving the original problem with CPLEX. Considering that the CFL instances are of small size, they can be solved in a few seconds. Thus, we exclude the CFL instances from the numerical analyses conducted in the next sections so as to focus on more challenging problems.

Finally, considering the results obtained on the $r10$ instances, the LDD method reaches an optimality gap of 19.37% which is much tighter than that obtained by the BD method (45.22%) or our algorithms (that fail to produce an optimality gap). For these instances, the four proposed BDD variants require more that 1 hour of computation time to solve the LP relaxation. Thus, our algorithm terminates before generating any strengthened or Lagrangian cuts. However, if the time limit to solve the root node is increased, then we observe that the proposed BDD_3 and BDD_4 variants reach a much tighter bound than the LDD method, as reported in Table 5.4. We next specifically analyze the impact of generating strengthened

and Lagrangian feasibility cuts and consider how the upper bound obtained at the root node is affected by the use of the proposed BDD method.

Impact of the proposed feasibility cuts

The results reported for FMCND in Table 5.1 were obtained using the classical feasibility cuts. We now assess how the performance of our method is affected when the proposed strengthened and Lagrangian feasibility cuts are generated in the second and third phase of the algorithm. These numerical results are presented in Table 5.2. It should be noted that we have only presented the results for the BDD_1 and BDD_3 variants (the results obtained for BDD_2 and BDD_4 being very similar to them, respectively).

When analyzing the results in Table 5.2, it is clear that generating strengthened and Lagrangian feasibility cuts has less impact on tightening the root node bound, when compared to the inclusion of the proposed optimality cuts. This was to be expected given that feasibility cuts do not bound the θ variables in the master problem. Moreover, the generation of the proposed feasibility cuts increases the time requirements. For these reasons, a deterioration of the obtained root node gaps is observed. One can thus anticipate that the proposed feasibility cuts will be most beneficial for problems where finding feasible solutions is a challenge. However, given the marginal impact of these cuts in the present case, we henceforth only use the classical feasibility cuts to solve the FMCND instances.

Quality of the upper bound

We now assess the quality of the upper bounds obtained from solving MIP subproblems at the root node when the different BDD variants or the LDD method are used. The percentage gaps of the obtained upper bounds with respect to the optimal solutions are summarized in Table 5.3. It should be noted that the computation times required to obtain these upper

Table 5.2 Performance of the proposed feasibility cuts versus the classical ones

	Classical Feasibility Cuts				Proposed Feasibility Cuts			
	BDD_1		BDD_3		BDD_1		BDD_3	
	Gap(%)	Time(s.)	Gap(%)	Time(s.)	Gap(%)	Time(s.)	Gap(%)	Time(s.)
r04	19.52	49.46	1.64	900.98	19.36	43.84	1.50	892.88
r05	13.17	239.90	1.14	2160.62	13.87	228.36	1.00	2281.52
r06	14.63	1026.42	9.72	2977.36	14.45	1004.79	10.18	3055.83
r07	16.50	70.10	3.58	1048.47	16.21	74.50	4.18	1034.03
r08	17.35	376.39	7.12	2673.75	17.45	303.84	7.55	2667.76
r09	16.99	1683.00	12.81	3681.55	16.74	1567.31	12.90	3842.33
r10	-	-	-	-	-	-	-	-
Mean	**16.36**	**574.21**	**6.00**	**2240.46**	**16.35**	**537.11**	**6.22**	**2295.72**

bounds were reported in Table 5.1. Also, we again recall that, in the case of the $r10$ instances, no results are reported for the different BDD variants given that the methods were unable to solve the LP relaxations in the maximum allotted time of 1 hour.

Table 5.3 Average percentage gap of the upper bound obtained by different method from the optimal value

Prob.	Inst.	Dual decomposition	The proposed BDD method			
			BDD_1	BDD_2	BDD_3	BDD_4
FMCND	r04	2.03	24.86	27.70	0.77	3.09
	r05	1.74	18.86	22.41	0.36	5.08
	r06	2.00	15.13	15.80	3.61	13.58
	r07	3.06	29.16	29.99	2.06	5.65
	r08	3.29	35.48	35.29	2.59	2.93
	r09	5.32	26.01	23.89	4.84	20.79
	r10	(22.44)	-	-	-	-
	Mean	**2.91**	**24.92**	**25.85**	**2.37**	**8.52**
CFL-S	101-104	1.34	6.69	6.69	0.07	0.23
	111-114	5.37	15.66	15.66	0.06	0.18
	121-124	8.61	14.27	14.27	0.08	0.15
	131-134	19.00	15.36	15.36	1.41	0.28
	Mean	**8.58**	**13.00**	**13.00**	**0.40**	**0.21**
SNI	3-30	15.82	24.47	22.71	2.94	3.72
	3-40	20.54	28.78	34.13	6.44	6.44
	3-50	27.61	40.92	39.50	8.90	7.93
	3-60	36.22	46.61	47.13	11.36	11.36
	3-70	43.92	44.63	44.41	16.89	16.89
	3-80	54.33	46.88	46.43	12.59	12.78
	3-90	57.66	47.25	42.23	15.22	19.93
	4-30	18.71	37.05	37.05	3.58	5.92
	4-40	26.89	51.27	51.27	6.46	6.88
	4-50	34.27	61.00	56.89	11.65	11.99
	4-60	44.03	78.67	78.67	13.55	13.55
	4-70	57.01	78.01	78.01	13.75	14.12
	4-80	75.08	82.48	82.48	16.87	17.16
	4-90	88.26	76.89	76.89	15.61	16.19
	Mean	**42.88**	**53.21**	**52.70**	**11.13**	**11.78**

From Table 5.3, we observe that the obtained upper bounds by the BDD_3 variant are very close to the optimal solutions. When compared to BDD_4, BDD_3 finds better upper bounds, which can be explained by the fact that, in this variant, the integer requirements are imposed on all the copied variables in the subproblems. It thus generates more integer solutions. Similarly, given their respective definition, BDD_3 and BDD_4 yield a much larger pool of integer solutions when compared to the BDD_1 and BDD_2 variants. Consequently, these variants greatly improve the quality of the upper bounds generated. Moreover, the upper bounds obtained by BDD_3 and BDD_4 (with the exception of the FMCND instances) are much better than those generated by the LDD method, which clearly indicates that our method provides an improved search mechanism. Last but not least, the quality of the primal

bounds reported in Table 5.3 shows that the proposed enhancements significantly alleviate the primal inefficiencies of the BD method.

5.6.2 Computational results with branch-and-cut

As previously observed, our method increases the time that is spent at the root node of the branch-and-bound tree. Thus, to resolve the issue of whether or not it is computationally beneficial to apply the proposed method, we now compare the variants BDD_3 and BDD_4 to both the BD and LDD methods by running the algorithms for a time limit of 10 hours (where at most 5 hours are dedicated to solve the LP at the root node). These numerical results are summarized in Table 5.4 where, in addition to the average total running times and gaps obtained upon completion, the number of solved instances within an optimality gap of 1% (i.e., column #Sol.) are also reported.

From Table 5.4, we observe that our algorithms reach much better optimality gaps and solve more instances in noticeably shorter CPU times. Superiority of the proposed method compared to the classical BD algorithm is without exception. These results are explained by the fact that our method generates considerably smaller branch-and-bound trees even though a larger amount of time is spent at the root node. Moreover, the LDD method (due to its slow progression) fails to reach to the same solution quality than both BDD_3 and BDD_4 obtain at the root node (see Tables 5.1 and 5.3).

When comparing the two proposed variants, one observes that BDD_3 performs better than BDD_4 when solving the ND and FL instances. These results can be explained by two main reasons. First, when the BDD_3 variant is applied, stronger cuts are generated and, due to the integrality requirements being imposed on all the copied variables in the subproblems, a more extensive pool of integer solutions is obtained. Therefore, tighter lower and upper bounds are generated at the root node (see Tables 5.1 and 5.3). The second reason explaining these results is that, due to the proposed variable fixing strategy, the time requirements to solve a subproblem in both BDD_3 and BDD_4 are quite comparable as a consequence of fixing the indicator variables. Finally, we observe that BDD_4 outperforms BDD_3 on the SNI instances. However, the explanation for this is that the variable fixing strategy, in this case, cannot be applied due to the presence of a budget constraint. Therefore, when solving the SNI instances, there is an added time requirement that is needed to solve the subproblems when applying the BDD_3 variant, when compared to BDD_4.

Table 5.4 Comparing the proposed decomposition method to the classical primal and dual decomposition methods

Prob.	Inst.	Benders decomposition			Dual decomposition			BDD_3			BDD_4		
		Time(s.)	Gap(%)	#Sol.	Time(s.)	Gap(%)	#Sol.	Time(s.)	Gap(%)	#Sol.	Time(s.)	Gap(%)	#Sol.
FMCND	r04	6045.47	1.08	5/5	10946.13	2.84	4/5	1233.76	0.88	5/5	952.08	1.06	5/5
	r05	21836.23	8.40	2/5	22376.22	1.90	2/5	1994.71	0.81	5/5	2145.48	0.58	5/5
	r06	29114.44	11.27	1/5	24045.43	8.30	2/5	17156.79	4.04	3/5	16498.10	3.03	3/5
	r07	25743.46	6.03	2/5	22853.89	5.54	3/5	4956.88	0.96	5/5	5350.06	0.98	5/5
	r08	36315.18	12.20	1/5	23755.06	7.59	2/5	9188.43	1.57	4/5	8840.45	1.33	4/5
	r09	30972.97	14.54	2/5	30686.53	13.81	1/5	23780.16	6.69	3/5	21000.60	5.59	3/5
	r10	31445.58	15.73	1/5	33104.18	16.96	1/5	33175.08	11.23	2/5	34025.04	11.10	1/5
	Mean	**25924.76**	**9.89**	**14/35**	**23966.78**	**8.14**	**15/35**	**13069.40**	**3.74**	**27/35**	**12687.40**	**3.38**	**26/35**
CFL-S	101-104	36796.25	14.62	0/4	5292.66	1.05	4/4	1100.41	0.75	4/4	645.68	0.64	4/4
	111-114	36254.33	8.20	0/4	28951.78	2.27	1/4	2489.96	0.94	4/4	2739.68	0.99	4/4
	121-124	36227.73	17.66	0/4	18430.40	1.08	4/4	4278.08	1.04	4/4	4799.71	1.04	4/4
	131-134	36294.90	19.87	0/4	31224.45	1.19	2/4	18803.08	1.10	3/4	14031.66	1.26	3/4
	Mean	**36393.30**	**15.09**	**0/16**	**20974.82**	**1.40**	**11/16**	**6667.88**	**0.96**	**15/16**	**5554.18**	**0.98**	**15/16**
SNI	3-30	2762.35	1.10	4/5	36471.92	14.67	0/5	2121.51	0.97	5/5	1936.05	1.07	5/5
	3-40	30982.14	1.80	0/5	36534.46	18.95	0/5	8639.26	1.10	4/5	9424.96	1.10	4/5
	3-50	27298.87	2.47	2/5	36381.64	23.25	0/5	6535.48	1.10	4/5	7963.64	1.06	4/5
	3-60	29300.58	2.66	0/5	36466.10	24.93	0/5	4693.26	1.10	4/5	4795.40	1.03	5/5
	3-70	32136.92	4.54	0/5	36460.78	29.92	0/5	3896.33	1.06	4/5	4493.22	1.04	5/5
	3-80	33614.04	5.32	1/5	36620.18	34.55	0/5	8982.22	1.10	2/5	19887.93	1.14	2/5
	3-90	36669.58	8.06	0/5	36496.58	39.25	0/5	16600.52	1.52	3/5	18786.01	1.54	3/5
	4-30	2253.55	1.10	3/5	36389.18	15.82	0/5	3060.10	1.10	4/5	2219.62	1.06	5/5
	4-40	12231.08	1.10	1/5	36441.30	18.89	0/5	6582.58	1.10	2/5	6281.24	1.10	4/5
	4-50	20818.73	2.16	2/5	36291.34	24.05	0/5	7417.27	0.69	5/5	7212.67	1.10	4/5
	4-60	21378.10	2.40	1/5	36459.86	28.04	0/5	10257.96	1.10	2/5	9263.12	0.90	4/5
	4-70	22587.84	1.72	3/5	36493.82	32.99	0/5	10466.20	0.89	3/5	6048.30	1.10	5/5
	4-80	25946.64	1.28	2/5	36648.72	38.75	0/5	4081.00	1.06	5/5	4607.59	0.77	4/5
	4-90	21177.38	1.74	1/5	36321.86	41.89	0/5	7831.69	1.10	5/5	14077.00	2.06	3/5
	Mean	**22796.98**	**2.67**	**20/70**	**36462.70**	**27.57**	**0/70**	**7226.10**	**1.07**	**52/70**	**8356.91**	**1.15**	**57/70**

5.6.3 Comparison with a state-of-the-art optimization solver

In this last section, we will assess how our proposed method (i.e., specifically the BDD_3 and BDD_4 variants) compares with a state-of-the-art optimization solver (i.e., CPLEX 12.7). To do so, we add a family of classical inequalities to the subproblems of the FMCND and CFL-S instances to provide comparable results to CPLEX which extensively exploits the special structures of these classical problems. These inequalities, which are redundant in the original models but may help to strengthen the relaxations, take the form of $x_a^k \leq \min\{d^k, u_a\}y_a$, for the arcs a in the network. They state that the amount of flow on each connecting arc a for each customer k has to be lower than the minimum between the customer's demand d_k and the arc's capacity u_a. It should be noticed that the use of these inequalities does not yield any complication in our implementations given the presence of both the integer and continuous variables in the subproblems. Moreover, they do not require a separation and lifting procedure. However, these inequalities are not added to the extensive formulation, there is an exponential number of them and their inclusion noticeably slows down the CPLEX solver. The results for this numerical comparison are provided in Table 5.5. We again report the average running times, optimality gaps and the number of solved instances by each method.

Table 5.5 Comparing the proposed method to the state-of-the-art optimization solver

Prob.	Inst.	CPLEX			BDD_3			BDD_4		
		Time(s.)	Gap(%)	#Sol.	Time(s.)	Gap(%)	#Sol.	Time(s.)	Gap(%)	#Sol.
FMCND	r04	258.10	0.57	5/5	410.57	0.44	5/5	385.64	0.71	5/5
	r05	1718.28	0.72	5/5	1485.92	0.56	5/5	1168.67	0.56	5/5
	r06	16076.07	2.92	3/5	12803.33	0.90	5/5	4897.01	0.61	5/5
	r07	699.17	0.89	5/5	1384.93	0.69	5/5	1231.60	0.76	5/5
	r08	8529.26	1.72	4/5	7430.11	0.65	5/5	7392.03	0.66	5/5
	r09	21741.47	8.28	3/5	17370.16	1.67	4/5	11961.41	1.57	4/5
	r10	28973.48	11.52	1/5	28950.08	8.65	1/5	28908.78	6.72	1/5
	Mean	**11142.26**	**3.80**	**26/35**	**9976.44**	**1.94**	**30/35**	**7992.16**	**1.66**	**30/35**
CFL-S	101-104	50.09	0.00	4/4	25.14	0.00	4/4	24.15	0.00	4/4
	111-114	1252.44	0.05	4/4	647.24	0.01	4/4	421.70	0.01	4/4
	121-124	2345.20	0.15	4/4	640.15	0.00	4/4	267.45	0.00	4/4
	131-134	1399.11	0.08	4/4	196.53	0.00	4/4	130.22	0.00	4/4
	Mean	**1261.71**	**0.07**	**16/16**	**377.27**	**0.00**	**16/16**	**210.88**	**0.00**	**16/16**
SNI	30	36225.72	7.81	0/5	2121.51	0.97	5/5	1936.05	1.07	5/5
	40	36168.22	15.60	0/5	8639.26	1.10	4/5	9424.96	1.10	4/5
	50	36163.76	19.33	0/5	6535.48	1.10	4/5	7963.64	1.06	4/5
	60	36166.40	19.22	0/5	4693.26	1.10	4/5	4795.40	1.03	5/5
	70	36155.20	17.61	0/5	3896.33	1.06	4/5	4493.22	1.04	5/5
	80	36155.84	20.14	0/5	8982.22	1.10	2/5	19887.93	1.14	2/5
	90	36150.18	24.25	0/5	16600.52	1.52	3/5	18786.01	1.54	3/5
	30	36183.46	12.00	0/5	3060.10	1.10	4/5	2219.62	1.06	5/5
	40	36147.56	17.94	0/5	6582.58	1.10	2/5	6281.24	1.10	4/5
	50	36141.76	22.39	0/5	7417.27	0.69	5/5	7212.67	1.10	4/5
	60	36136.64	24.92	0/5	10257.96	1.10	2/5	9263.12	0.90	4/5
	70	36138.64	42.29	0/5	10466.20	0.89	3/5	6048.30	1.10	5/5
	80	36129.26	29.25	0/5	4081.00	1.06	5/5	4607.59	0.77	4/5
	90	36133.82	60.29	0/5	7831.69	1.10	5/5	14077.00	2.06	3/5
	Mean	**36156.89**	**23.79**	**0/70**	**7226.10**	**1.07**	**52/70**	**8356.91**	**1.15**	**57/70**

Before analyzing the results reported in Table 5.5, a general comparison between these results and those provided in Table 5.4 shows that a simple and straightforward use of classical valid inequalities noticeably improves the performance of our method. The overall insight that can be gained from this comparison is that the bulk of the literature on the acceleration techniques that have been developed for the BD method (Rahmaniani et al., 2017a) can be applied here also and one can expect that this would noticeably enhance the numerical performance of the proposed method.

As for the specific results provided in Table 5.5, one first observes that the only cases where CPLEX performs better are when it is used to solve the $r04$ and $r07$ instances (i.e., CPLEX solves these instances in less than 10 minutes). For the rest of the instances, CPLEX is not competitive with either the BDD_3 and BDD_4 variants, both of which optimally solve a larger subset of the instances and in much shorter computation times. Moreover, CPLEX fails to solve any of the SNI instances after 10 hours of computational effort, while the two

variants of our proposed method solve more that 74% of these instances in approximately 2 hours. Finally, when directly comparing the two developed variants, one finds that BDD_4 generally outperforms BDD_3 (in terms of the average computation times, gaps obtained and number of instances solved). Overall, this is explained by the fact that the BDD_4 variant is able to solve the root node faster, albeit while generating weaker cuts when compared to BDD_3. However, the inclusion of the valid inequalities mitigates the latter disadvantage of BDD_4 by reducing the overall differences observed between the lower bounds provided at the root node by the two variants. In turn, when the branch-and-bound process begins, the time reductions at the root node obtained via BDD_4 become the determining factor in providing a more efficient solution process for our proposed method.

5.7 Conclusions

In this paper, we have proposed a decomposition method that combines the complementary advantages of the classical Benders and Lagrangian dual decomposition approaches. It generates cuts that dominate the classical optimality and feasibility cuts at fractional points of the master problem. The proposed method also tightens the LP relaxation of the MP problem, which we have shown to be at least as tight as the best bound obtained by the Lagrangian decomposition method. Another important feature of our method lies in its enhanced capabilities to find high quality incumbent solutions at early iterations of the algorithm.

We have applied the proposed method to solve a wide range of hard combinatorial optimization instances. It was observed that, for a reasonable time limit, the proposed method was able to reach much tighter root node bounds when compared to the Lagrangian dual decomposition approach. Our algorithm was also capable of finding incumbent solutions very close to the optimal values at the early iterations of the search process. Furthermore, it was numerically shown that the developed method increases the time spent on the root nodes when compared to the classical Benders decomposition method. However, this added effort, which produces significantly tighter bounds at the root nodes, is largely compensated by the fact that the improved bounds then enable the algorithm to generate smaller branch-and-bound trees to solve the instances. Finally, we observed that our method also outperforms a state-of-the-art optimization solvers.

Going forward, there are many avenues of research to further improve the proposed algorithm. When applied on stochastic models, one could first take advantage of parallel computing to solve the subproblems. Furthermore, as originally presented, when our algorithm is used to solve a stochastic model, each subproblem is defined using a single scenario. However, it has been previously shown that having subproblems defined on clusters of scenarios yields both

tighter lower bounds and better incumbent solutions. Thus, exploring how different clustering strategies applied on scenarios would impact the overall performance of our proposed method is definitely an interesting line of inquiry. In addition, there have been numerous studies dedicated to the improvement of both the Benders and Lagrangian decomposition methods. Investigating which of these strategies could be incorporated in our method to further enhance its numerical performance would be of interest. For example, we made use of a simple cutting plane method to update the Lagrangian multipliers. In this regard, more effective strategies could be employed to significantly improve the proposed algorithm. Last but not least, the proposed method guarantees convergence even when the Benders subproblems are nonlinear mixed-integer problems. Thus, numerically assessing how the proposed method would fare against classical algorithms, such as the integer L-shaped and outer approximation methods, is certainly worthwhile.

CHAPTER 6 ARTICLE 4: THE ASYNCHRONOUS BENDERS DECOMPOSITION METHOD

Ragheb Rahmaniani[1], Teodor Gabriel Crainic[2], Michel Gendreau[1], Walter Rei[2]

[1] CIRRELT & Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal H3C 3A7, Canada.

[2] CIRRELT & School of Management, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal H3C 3P8, Canada.

Abstract. The Benders decomposition method is a widely used approach in addressing stochastic integer programs with recourse. The available parallel variants of this method are based on a synchronized master-slave implementation in which the slave processors wait until the master problem is solved, and vice versa. This parallelization may, however, suffer from significant load imbalance, particularly when an iteration of the master problem can take hours. We thus propose parallelization strategies for the Benders decomposition algorithm in a branch-and-cut framework. To further reduce the idle times, we relax the synchronization requirements among the master and slave processors. However, the combination of the asynchronous communications and branch-and-cut implementation results in an algorithm for which we are unable to prove its global convergence and it may even underperform the sequential algorithm. Therefore, we study the convergence of the algorithm and propose various acceleration strategies to obtain an effective parallel method. We conduct an extensive numerical study on benchmark instances from stochastic network design problems. The results indicate that our asynchronous algorithm reaches higher speedup rates compared to the conventional parallel methods. We also show that it is several orders of magnitude faster than the state-of-the-art solvers.

Keywords. *Benders decomposition*, *Branch-and-cut*, *Stochastic integer programming*, *Parallel computing.*

History. Submitted to INFORMS Journal on Computing.

6.1 Introduction

Stochastic integer programming (SIP) offers a powerful tool to deal with uncertainties in planning problems where the distribution of the uncertain parameters is assumed to be known and often characterized with a finite set of discrete scenarios (Birge and Louveaux, 1997). We consider here the special two-stage case of SIPs where decisions are made in two

stages. In the *first-stage* the decision maker must make a decision now, while not knowing the exact outcome of the uncertain parameters. In the *second-stage* when the parameters have become known, the decision maker can take recourse actions to adjust his/her plan accordingly.

SIP models are usually very large in size and very difficult to solve due to the data uncertainty and their combinatorial nature (Ahmed, 2010). However, they exhibit special structures amenable to decomposition methods (Ruszczyński, 1997). Therefore, efforts have been made to design various decomposition-based algorithms for these problems, e.g., Benders decomposition (BD) (Benders, 1962) also known as L-shaped method (Van Slyke and Wets, 1969), stochastic decomposition (Higle and Sen, 1991), nested decomposition (Archibald et al., 1999), subgradient decomposition (Sen, 1993), scenario decomposition (Rockafellar and Wets, 1991), disjunctive decomposition (Ntaimo, 2010), etc. Among these methods, the BD has become a prominent methodology to address stochastic programs with recourse (Ruszczyński, 1997; Uryasev and Pardalos, 2013).

In the BD method the model is projected onto the subspace defined by the first-stage variables. The projected term is then replaced by its dual counterpart. Accordingly, an equivalent model is built by iteratively enumerating the extreme points and rays of the dual program, referred to as the *subproblem* (SP). At each iteration, one solves a *master problem* (MP), which initially includes no constraints except for those imposed on the first-stage variables. The obtained solution is fixed in the SP. If the resulting SP is feasible, the corresponding optimal extreme point is used to generate an *optimality cut*. Otherwise, an extreme ray is extracted to generate a *feasibility cut*. The generated cut is then added to the MP and this process repeats until an optimal solution is found. If the MP is a mixed-integer program, the algorithm is commonly cast into a *branch-and-cut* (B&C) framework in order to avoid solving an integer problem from scratch at each iteration. Thus, a single branch-and-bound tree is built and the cuts are generated at the integer (and possibly some fractional) nodes (Rahmaniani et al., 2017a).

The SP decomposes by scenario. Solving these scenario SPs at each iteration usually corresponds to the most time-consuming part of the algorithm, because a large number of scenarios are often required to properly set the value of uncertain parameters. These SPs are disjoint and can be solved in parallel. Thus, *parallel computing* appears very promising to effectively accelerate solution of SIP problems when the BD method is used (Linderoth and Wright, 2003; Li, 2013).

Although parallel processing saves time, the processors at some point require to exchange information and consolidate the results to create work units for next iteration. In the parallel

BD methods of the literature, these points are implemented using synchronized communications among the processor that solves the MP and the processors that solve the SPs. This, however, causes having one or several idle processors at any given time, particularly when an iteration of the MP can take hours (Yang et al., 2016). In this case, the efficiency of the parallel execution may decrease as the number of the processors increases. It is thus important to design new parallelization schemes for the BD method to obtain a high-performing algorithm for the SIPs.

In this article we aim at developing effective parallelization strategies for the B&C implementation of the BD method. To the best of our knowledge, this is the first article to consider parallelization of the BD method in a B&C framework. It is important to note that in this case, the resulting parallel algorithm is different from the existing parallel B&C algorithms. In the parallel B&C methods the main emphasis is on parallelizing the branch-and-bound tree and the cuts are only to accelerate the convergence (Ralphs et al., 2003; Crainic et al., 2006b). While in the parallel BD methods the main emphasis is on parallelizing the SPs and the cuts are necessary for the convergence. Thus, the parallelization strategies of the B&C algorithms cannot easily be translated into parallel BD methods and, we focus on the strategies in which the SPs are optimized in parallel on various *slave* processors and the MP is solved sequentially on a single *master* processor.

To realize our goal, we relax the synchronization requirements among the master and slave processors. This means that the algorithm waits only for a small portion of the cuts at each (integer) node of the branch-and-bound tree. Although this significantly reduces the idle times, it results in an algorithm for which we are unable to prove its global convergence. This happens because, in absence of a rigid synchronization, the necessary cuts at the integer nodes may not be generated at the right moment. We thus study this issue and show that with an appropriate B&C design the algorithm can converge. On the other hand, the asynchronous algorithm may execute a large amount of *redundant work* since the MP executes its next iteration with a *partial* feedback from the SPs. This can cause serious efficiency issues such that the algorithm may even underperform the sequential variant. Therefore, we propose various acceleration techniques to obtain an efficient asynchronous BD algorithm. The main contributions of this article are thus severalfold :

- Proposing an effective Benders-based *asynchronous* parallel B&C algorithm for two-stage SIP problems. We also present the *synchronized* algorithm and an *hybrid* of the two algorithms ;
- Studying the convergence of these algorithms and proposing strategies to accelerate their numerical performance. In this regard, we revisit some of the classical acceleration techniques to properly fit into our parallel frameworks and we propose novel ones ;

— Presenting extensive numerical results on benchmark instances to assess the proposed strategies and algorithms. We provide guidelines on how to properly use the parallelization techniques and discuss various fruitful research directions.

The remainder of this article is organized as follows. In section 6.2, the problem of interest and the sequential BD algorithm are presented. In section 6.3, we classify and review parallel BD methods. We present our synchronized, asynchronous, and hybrid parallel algorithms in sections 6.4, 6.5, and 6.6, respectively. We discuss the implementation details and numerical results in sections 6.7 and 6.8. Finally, conclusions and future remarks are summarized in the last section.

6.2 The Benders decomposition method

In this section, we first recall the two-stage stochastic problem of interest. Then, we present a sequential BD algorithm to solve it.

6.2.1 Two-stage stochastic integer programming

In two-stage stochastic programming, the uncertainty is observed only once and decisions are made before and after observing the uncertainties. The common practice is to approximate the probability distribution of random variables by a discrete probability distribution with a finite support. This gives a finite set of scenarios, each representing a possible realization of the random events. Given the scenario set $\mathcal{S}$ and occurrence probability $\rho_s > 0$ for each $s \in \mathcal{S}$ such that $\sum_{s\in\mathcal{S}} \rho_s = 1$, a canonical representation of a two-stage stochastic program is :

$$z^* := \quad \min_y \{f^\top y + \sum_{s\in\mathcal{S}} \rho_s Q(y,s) : \ By \geq b, \ y \in \mathcal{Y}\} \tag{6.1}$$

where for each scenario $s \in \mathcal{S}$

$$Q(y,s) = \quad \min_x \{c_s^\top x : W_s x \geq h_s - T_s y, \ x \in \mathcal{X}\} \tag{6.2}$$

with $f \in \mathbb{R}^n$, $B \in \mathbb{R}^{k\times n}$, $b \in \mathbb{R}^k$, $c_s \in \mathbb{R}^m$, $W_s \in \mathbb{R}^{l\times m}$, $h_s \in \mathbb{R}^l$, $T_s \in \mathbb{R}^{l\times n}$. Here, $\mathcal{X} \subseteq \mathbb{R}^m$ and $\mathcal{Y} \subseteq \mathbb{R}^n$ are nonempty closed subsets which define the nature of the x and y decision variables in terms of sign and integrality restrictions. In this article we assume that $\mathcal{Y}= \mathbb{Z}^n_+$ and $\mathcal{X}= \mathbb{R}^m_+$. In this program, y represents the first-stage decisions and x represents the second-stage decisions. We thus seek a feasible solution that minimizes the first-stage cost $f^\top y$ plus the expected cost of the second-stage decisions.

6.2.2 Sequential Benders decomposition method

For a tentative value of the first-stage variables $\bar{y}$, the recourse problem $Q(\bar{y}, s)$ is a continuous linear program. Given a dual variable α associated with constraint $W_s x \geq h_s - T_s \bar{y}$, the dual of $Q(\bar{y}, s)$ is

$$(SP(\bar{y}, s)) \;\; Q(\bar{y}, s) = \max_{\alpha} \{(h_s - T_s \bar{y})^\top \alpha : \; {W_s}^\top \alpha \leq c_s, \; \alpha \in \mathbb{R}^l_+\} \tag{6.3}$$

The above program is either unbounded or feasible and bounded. In the former case, the $\bar{y}$ solution is infeasible and thus there exists a direction of unboundedness $r_{q,s}$, $q \in F_s$ that satisfies $(h_s - T_s y)^\top r_{q,s} > 0$, where F_s is the set of extreme rays of (6.3). To assure the feasibility of the y solutions, we need to forbid the directions of unboundedness through imposing $(h_s - T_s y)^\top r_{q,s} \leq 0, \; q \in F_s$, on the y variables, which gives

$$z^* \; = \; \min_{y \in \mathbb{Z}^n_+} \{f^\top y + \sum_{s \in \mathcal{S}} \rho_s Q(y, s) : \; By \geq b, \; (h_s - T_s y)^\top r_{q,s} \leq 0 \; \forall s \in \mathcal{S}, q \in F_s\} \tag{6.4}$$

In the latter case, the optimal solution of the SP is one of its extreme points $\alpha_{e,s}, \; e \in E_s$, where E^s is the set of extreme points of (6.3). We can thus rewrite program (6.4) in an extensive form by interchanging $Q(y, s)$ with its dual, i.e., $SP(y, s)$

$$\min_{y \in Y} \Big\{ f^\top y + \sum_{s \in S} \rho_s \max_{e \in E_s} \{(h_s - T_s y)^\top \alpha_{e,s}\} : \; (h_s - T_s y)^\top r_{q,s} \leq 0 \;\; s \in S, q \in F_s \Big\} \tag{6.5}$$

where $Y = \{y : By \geq b, y \in \mathbb{Z}^n_+\}$. If we capture value of the inner maximization in a single variable θ_s for every $s \in \mathcal{S}$, we can obtain the following equivalent reformulation of (6.1), called Benders *master problem* (MP) :

$$MP(E_1, ..., E_{|\mathcal{S}|}; F_1, ..., F_{|\mathcal{S}|}) \; := \; \min_{y \in Y} f^\top y + \sum_{s \in S} \rho_s \theta_s \tag{6.6}$$

$$(h_s - T_s y)^\top \alpha_{e,s} \leq \theta_s \qquad s \in S, e \in E_s \tag{6.7}$$

$$(h_s - T_s y)^\top r_{q,s} \leq 0 \qquad s \in S, q \in F_s \tag{6.8}$$

Enumerating all the extreme points E_s and extreme rays F_s for each SP $s \in S$ is computationally burdensome and unnecessary. Thus, Benders (1962) suggested a delayed constraint generation strategy to generate the *optimality* (6.7) and *feasibility* (6.8) cuts on the fly. In the classical BD method, the MP is solved from scratch at each iteration. However, nowadays, this method is commonly implemented in a B&C framework in order to avoid solving a MILP MP at each iteration (Naoum-Sawaya and Elhedhli, 2013; Rahmaniani et al., 2017b). Moreo-

ver, the LP relaxation of the MP is often solved first to quickly obtain valid cuts and tighten the root node which enables the Benders method to perform more efficiently (McDaniel and Devine, 1977). Algorithm 2 presents the pseudo-code of this algorithm.

Algorithm 2 : The sequential Benders decomposition algorithm

1: Create the MP which is the LP relaxation of program (6.6)-(6.8) with $E_s = \emptyset$ and $F_s = \emptyset$, $\forall s \in \mathcal{S}$
2: Iteratively solve the MP and add cuts
3: Add the obtained MP into tree L, set $UB = \infty$, $LB = -\infty$, and ϵ_{opt} to the optimality tolerance
4: **while** $UB - LB > \epsilon_{opt}$ **and** $L \neq \emptyset$ **do**
5: Select a node from L
6: Solve this node to get an optimal solution $\bar{y}$ with objective value of ϑ^*
7: **if** *node was infeasible* **or** $\vartheta^* \geq UB$ **then**
8: Prune the node and go to line 5
9: **end if**
10: **if** $\bar{y}$ *is integer* **then**
11: **while** *a violating cut can be found* **and** *the* $\bar{y}$ *is integer* **do**
12: **for all** $s \in \mathcal{S}$ **do**
13: Solve $SP(\bar{y}, s)$
14: **if** $SP(\bar{y}, s)$ *was infeasible* **then**
15: Add a feasibility cut and go to line 20
16: **else**
17: Extract the optimal extreme point e_s
18: **end if**
19: **end for**
20: Add new optimality cut(s) using the extreme points obtained in line 17
21: Solve this node again to get an optimal solution $\bar{y}$ with objective value of ϑ^*
22: **if** *node was infeasible* **or** $\vartheta^* \geq UB$ **then**
23: Prune the node and go to line 5
24: **end if**
25: **end while**
26: **if** $\bar{y}$ *is integer* **then**
27: Set $UB = \min\{UB, \vartheta^*\}$, prune the node, and go to line 5
28: **end if**
29: **end if**
30: Set LB as the minimum objective values of the nodes in L
31: Choose a fractional variable from $\bar{y}$ to branch
32: Create two nodes and add them to L.
33: **end while**

The algorithm is applied in two phases. It first starts with solving the LP relaxation of the MP which initially includes no optimality and feasibility cuts (lines 1 and 2). Thus, for a

certain number of iterations, it sequentially solves the MP and SPs to generate optimality and feasibility cuts. In the second phase, a branch-and-bound tree is created (line 3), where the root node is the MP with the generated cuts at the first phase. At each iteration, the algorithm selects and solves a node from the pool L (lines 5 and 6). If the obtained solution is fractional and the node cannot be pruned (line 7), a branching occurs to create and add two new nodes to the pool (lines 24 and 25). If the node cannot be pruned (line 7) and its solution is integer (line 9), the algorithm iteratively adds cuts until it can either prune that node, the solution becomes fractional, or no more violated cut can be found (lines 10 to 20). This process (lines 5 to 25) repeats until the bounds collide or the node pool becomes empty. Note that in line 22 the upper bound is updated, i.e., the potential incumbent is accepted, if the $\bar{y}$ is an integer solution that satisfies all the Benders cuts.

Our developments in this article are based on Algorithm 2, which we refer to as the *Branch-and-Benders-cut* (B&BC) method. Various studies have been developed to accelerate the BD algorithm. To avoid burdening this article, we refer the reader to Rahmaniani et al. (2017a) for a complete treatment of this topic.

6.3 Parallelization strategies and previous work

The BD method lends itself readily to parallelization as it creates a MP and many disjoint SPs. Thus, efforts have been made to take advantage of parallel computing in accelerating this method. To the best of our knowledge, all the existing parallel variants of this method follow the master-slave parallel programming model. We classify and review such methods in this part.

6.3.1 Master-slave parallelization strategies

The existing parallel BD method can be summarized as follows : *The MP is assigned to a processor, the "master", which also coordinates other processors, the "slaves", which solve the SPs. At each iteration, the solution obtained from solving the MP is broadcast to the slave processors. They then return the objective values and the cuts obtained from solving the SPs to the master and the same procedure repeats.* Such master-slave parallelization schemes are known as *low-level parallelism* as they do not modify the algorithmic logic or the search space (Crainic and Toulouse, 1998).

In many cases, the number of the processors is less than the number of the SPs and some SPs may be more time consuming than others. Therefore, it is important to take into account how work units are allocated to the processors to avoid having idle processors. To create work

units for next iteration, information must be exchanged among the processors. In a parallel environment, this necessitates having some sort of communication to share information. The type of communications strongly influences the design of parallel BD algorithms. For example, if communications are synchronized, the only difference between the parallel algorithm and the sequential one lies in solving the SPs in parallel. When communications are asynchronous, the processors are less interdependent. After broadcasting each MP solution, the master processor waits only for a subset of the cuts before re-optimizing the MP. For this reason, some SPs may remain unsolved which results in having a pool of unevaluated SPs. On the other hand, the slave processors continuously evaluate SPs. As a result, many cuts might be available when the master processor requests cuts. Therefore, it is important to decide : when to solve the MP, which solution to use in generating cuts, which SP to solve now, which cut to apply to the MP ? We thus define a three-dimension taxonomy, depicted in Figure 6.1, that captures all these factors.

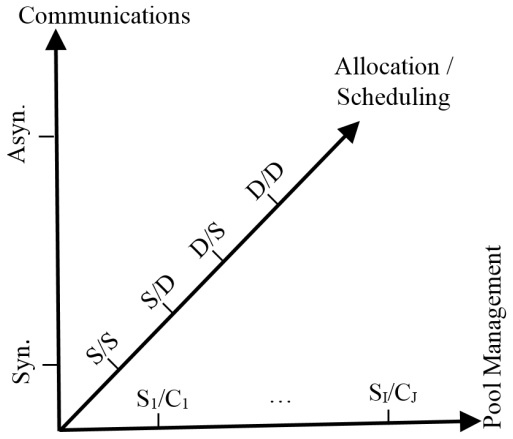

Figure 6.1 Taxonomy of the master-slave parallel Benders decomposition methods

— ***Communication*** : defines whether the processors at each iteration stop to communicate (*synchronous*) or not (*asynchronous*) ;
— ***Allocation/Scheduling*** : determines how the SPs and the MP are assigned to the processors and when they are solved. These decisions can be made either in a *dynamic* (D) or *static* (S) fashion. In the former, the decisions regarding when and where to solve each problem are taken during the course of the algorithm. In the latter, the decisions are made beforehand. Thus, S/D, for example, means static allocation and dynamic scheduling ;
— ***Pool Management*** implies the strategies used to manage the pool of solutions (denoted by $S_1, ..., S_I$) and the pool of cuts (denoted by $C_1, ..., C_J$), where I and J are the number of possible strategies to manage each pool, respectively.

A combination of various alternatives for these components yields a parallel BD algorithm. Proper strategies in each dimension need to be defined such that the overall *idle times* and amount of *redundant work* is minimized.

6.3.2 Previous work

Although parallelization of the BD method seems natural, the number of existing parallel variants of this method are very limited. This is due to the interdependency of MP and SPs, i.e., MP needs feedback from SPs before being able to execute its next iteration and vice versa. Moreover, due to the dependency of this framework on having multiple SPs, most of the parallel BD algorithms are developed for stochastic problems. However, since solving SPs is significantly time consuming, it has proven effective in various cases (Ariyawansa and Hudson, 1991; Wolf and Koberstein, 2013; Tarvin et al., 2016). The literature discusses some of the strategies and algorithmic challenges arising from this parallelization approach.

Dantzig et al. (1991) considered a dynamic work allocation strategy in which the next idle processor gets the next SP based on a first-in-first-out strategy until all the SPs are solved and the MP can be recomputed with the new cuts. The efficiency of this parallel algorithm did not exceed 60% even on machines with 64 processors. Similarly, Li (2013) observed that dynamic work allocation is superior to static work allocation, because it reduces idle times and also it saves executing extra work. For example, if an SP is infeasible, the evaluation of the SPs remaining in the queue will be terminated and the feasibility cut generation scheme will be launched.

Nielsen and Zenios (1997) exploited the structural similarities of the SPs by applying an interior point algorithm on a fine-grained parallel machine. They decided to sequentially solve the MP to optimality from scratch at each iteration because it can easily be handled by means of an interior point method. Vladimirou (1998) implemented a partial-cut aggregation strategy to reduce the communication overheads.

In some studies the decomposition has been modified to better suit the parallelization scheme. Dempster and Thompson (1998) proposed a parallel nested algorithm for multi-stage stochastic programs. The authors used stage aggregation techniques to increase the size of the nodes and therefore the time spent on calculations relative to the time spent communicating between processors. In a similar spirit, Chermakani (2015) observed that when the number of SPs is considerably larger than the number of available processors, so that some SPs must be solved sequentially, it may be better to aggregate some of them. Latorre et al. (2009) modified the decomposition scheme for multi-stage stochastic programs such that the subsets of nodes assigned to each SP may overlap. This, unlike former studies, allows to noticeably reduce the

dependency among SPs at each iteration, because they can be solved at the same time.

All the reviewed studies implement a synchronized parallelism. Moritsch et al. (2001) proposed a prototype for an asynchronous nested optimization algorithm. However, no specific strategy or numerical results were presented. Linderoth and Wright (2003) implemented an asynchronous communication scheme in which the MP is re-optimized as soon as a portion of the cuts are generated. Testing this algorithm on LP stochastic programs with up to 10^7 scenarios, the authors observed a time reduction up to 83.5% on a computational grid compared to the sequential variant.

The speedup rates of the low-level parallelizations are often limited. Pacqueau et al. (2012) observed that solving SPs accounts for more than 70% of the total time requirement, while they merely observed an speedup ratio up to 45% with 4 processors. Furthermore, low-level parallelism improves the efficiency if the MP solving does not dominate the solution process. Yang et al. (2016) observed that the benefit of parallelism fades away with the scale of the problem because the computational effort is dominated by solving the MP.

Some of the common knowledge for the sequential algorithm may not apply to its parallel variants. For example, Wolf and Koberstein (2013) pointed out that the single-cut version benefits from parallelization more than the multi-cut version because more SPs need to be solved in the former case. They also observed that the single-cut method may outperform the multi-cut variant because it requires less cuts in general although it executes a larger number of iterations.

To conclude this part, we summarize the literature in Table 6.1. We observe that the literature

Table 6.1 Summary of the Benders-based parallel algorithms

Reference	Problem class		Implementation		Communication		Work allocation		Scheduling		Pool management	
	LP	MIP	classic	B&C	Syn.	Asyn.	dynamic	static	dynamic	static	Solution	Cut
Chermakani (2015)	+		+		+			+		+		
Dantzig et al. (1991)	+		+		+		+			+		
Dempster and Thompson (1998)	+		+		+			+		+		
Latorre et al. (2009)		+	+		+		+			+		
Li (2013)		+	+		+		+			+		
Linderoth and Wright (2003)	+		+			+		+	+		+	+
Moritsch et al. (2001)	+		+			+		+	+		+	
Nielsen and Zenios (1997)	+		+		+			+		+		
Pacqueau et al. (2012)		+	+					+		+		
Vladimirou (1998)	+		+		+			+		+		
Wolf and Koberstein (2013)	+	+	+		+		+			+		
Yang et al. (2016)		+	+		+			+		+		

on parallel BD algorithms is sparse. Few studies consider integer programs and no one has implemented the parallel algorithm in the B&C format. Note that here "classic" refers to solving the MP from scratch at each iteration as in the classical version of the algorithm. Moreover, only two studies consider asynchronous communications and these are developed

for linear continuous problems. Also, we realized that synchronized parallelism is suitable if the MP can be solved quickly and the SPs constitute the major computational bottleneck of the algorithm. Thus, the reviewed strategies are not directly applicable to develop an efficient parallelization of the BD method when an iteration of the MP can take hours.

6.4 Synchronized parallel Benders decomposition algorithm

In the parallel synchronous version of Algorithm 2, each time the MP finds a new solution to generate cuts, it is broadcast to all other processors. The master processor waits for the slave processors to solve their assigned SPs. When the current solution is evaluated, the cuts are added to the MP. Then, the master processor continues with the next iteration until it finds another appropriate solution to generate cuts for which the same process is repeated. Thus, the only difference of this parallel algorithm with the sequential one lies in solving the SPs in parallel, i.e., the for-loop in line 11 of Algorithm 1.

To assign the SPs to the processors, given $|\mathcal{P}|$ slave processors and $|\mathcal{S}|$ scenario SPs, we group SPs and assign each group to a processor such that each processors receives a roughly equal number of SPs. This follows from the fact that in two-stage stochastic programming, the SPs have usually the same level of resolution difficulty. Thus by equally distributing them among the slave processors, load-balancing strategies are not required. In addition, we try to assign similar SPs to a processor, because this helps to more effectively take advantage of the re-optimization tools of our LP solver. The similarity is measured as the Euclidean distance of random parameters.

6.4.1 Cut aggregation

When the number of SPs is larger than number of the first-stage variables, it is not numerically the best strategy to add to the MP a cut per SP as in line 17 of Algorithm 2 (Trukhanov et al., 2010). Thus, we group the SPs into $|\mathcal{D}|$ clusters using the *k-mean++* algorithm (Arthur and Vassilvitskii, 2007), where $\mathcal{D}$ is the set of clusters with cardinality $|\mathcal{D}|$. Note that the number of SPs in each cluster may not be equal. Then, for each cluster we define a single recourse variable and aggregate the generated cuts in that cluster into a single cut.

6.4.2 Upper bound from fractional points

The upper bounds obtained from fractional master solutions (e.g., line 2 of Algorithm 2) are not valid for the original problem. Due to the high importance of the upper bound value in pruning nodes of the branch-and-bound tree, we suggest a practical strategy to extract valid

incumbent values from fractional solutions. This strategy requires the following assumptions : (i) $f_i \geq 0$, (ii) for a given feasible $\bar{y}$, $\hat{y}$ is also feasible if $\hat{y}_i \geq \bar{y}_i$ for all $i \in \{1, ..., n\}$, and (iii) $Q(\bar{y}, s) \geq Q(\hat{y}, s), s \in \mathcal{S}$.

Proposition 6.1. *Given a feasible fractional master solution $\bar{y}$, i.e., $\bar{y} \in Y \bigcap_{s\in\mathcal{S}}\{W_s x \geq h_s - T_s\bar{y}$, for some $x \in \mathbb{R}^m_+\}$ and the associated recourse costs ν^*_s for every $s \in \mathcal{S}$, a global upper bound can be obtained from $f^\top\lceil\bar{y}\rceil + \sum_{s\in\mathcal{S}} \rho_s\nu^*_s$, where $\lceil\rceil$ is a roundup function.*

Proof. See Appendix M. □

6.4.3 Cut generation

Conventionally, generating cuts at fractional nodes of the B&BC algorithms, except for the root node, is ignored. This is, to a very large extent, due to the excessive time required to generate them (Botton et al., 2013). However, in parallel B&BC methods we have more computational power at disposal. Thus, we study some strategies to generate cuts at fractional nodes of the branch-and-bound tree that we refer to as *cut generation* strategies.

The first strategy is to generate Benders cuts at the fractional nodes. Accordingly, we generate cuts for the first Γ fractional nodes, where $0 < \Gamma \leq \infty$ and 0 being the root node for which we always generate cuts. There are two possibilities at each fractional node : (i) generate at most one round of cuts per node or (ii) call upon the cut generation module as long as the local bound at the current node improves by more than $\tau\%$.

The second class of strategies rounds the fractional solutions to derive integer ones which can be used to obtain valid upper bounds and cuts. Specifically : (i) round the fractional values to the closest integer point ; (ii) round upward any value greater than λ and downward otherwise, where λ is a positive digit in $[0, 0.5)$; (iii) solve a restricted MILP program to round the fractional values. This problem is derived by considering the extensive formulation for a small subset of scenarios and fixing variables to 1 if their current value is greater than $1 - \lambda$ and to 0 if it is less than $\hat{\lambda}$.

Many of the dispatched solutions to the slave processors are infeasible. Thus, the last strategy revolves around a repair heuristic to restore the feasibility of such solutions. In many applications feasibility of the solutions satisfies the following monotonicity property : given a feasible solution $\bar{y}$, $\hat{y}$ is as well feasible if $\hat{y}_i \geq \bar{y}_i$ for $i \in \{1, ..., n\}$. The repair heuristic thus solves a restricted version of the problem with objective function of $\triangle(\bar{y}) := \sum_{i\in\{1,...,n\}}(1 - \bar{y}_i)y_i$, where variables are bounded from below by their current value, i.e., $y \geq \bar{y}$, to maintain the structure of current solution and solve this problem faster.

6.4.4 Cut management

The B&BC method, particularly with the cut generation strategies, may generate many cuts which are not all worth adding to or keeping in the master formulation. Thus, a cut corresponding to a fractional solution is added to the master formulation if its relative violation (the absolute violation of the cut divided by the 2-norm of the cut coefficients) is at least 10^{-3}, otherwise it is discarded.

Cut removal can computationally be expensive as it disturbs our LP solver, i.e., CPLEX. Thus, we execute the following routine only following the termination of the first phase to identify the dominated cuts. At iteration t, cut $(h_s - T_s y)^\top \alpha_s^t$ is generated to primarily bound the recourse problem s at solution $\bar{y}^t$. Following a heuristic notion, if there is another generated cut $(h_s - T_s y)^\top \alpha_s^j$, such that $j \neq t$, that bounds the recourse variable θ_s tighter at $\bar{y}^t$, i.e., $h_s^\top(\alpha_s^j - \alpha_s^t) + (\alpha_s^t - \alpha_s^j)^\top T_s \bar{y}^t \geq 0$, it flags the possibility of removing cut $(h_s - T_s y)^\top \alpha_s^t$ without deteriorating the approximated value function. We apply the same procedure for the feasibility cuts. Note that executing this test for all $y \in Y$ would yield an exact method to identify dominated cuts (Pfeiffer et al., 2012). In the second phase, our solver shall remove any cut that might become a slack.

6.5 Asynchronous parallel Benders decomposition algorithm

The synchronization requirement between master and slave processors increases the overheads due to the excessive idle times. This is particularly evident when solving any of the problems is noticeably more time consuming than others. A natural way to overcome this issue is to loosen the synchronization requirement among the master and slave processors. To achieve this, the master processor is required to wait only for $\gamma|\mathcal{S}|\%$ of the cuts from slave processors before executing its next iteration (i.e., resolving the MP in the first phase or exploring the tree in the second phase), where $0 \leq \gamma \leq 1$. However, for any $\gamma < 1$, the B&BC method may fail to converge and also, it may increase the amount of redundant work such that the parallel algorithm might underperform the sequential algorithm. These are the main issues in developing an effective asynchronous parallel Benders algorithm which we address them in this section.

6.5.1 Convergence

To ensure convergence of the B&C implementation of the B&BC method, the inner while-loop in Algorithm 2 cannot be stopped prematurely. Otherwise, an incumbent solution might be accepted in line 22 which does not necessarily satisfy all the Benders cuts. For this reason,

the asynchronism may compromise the convergence of the B&BC method as it applies only a subset of the cuts at each iteration of the while-loop. In fact, the applied cuts (which can be zero cuts based on the γ value) may not affect the current integer solution and thus, resolving the associated node in line 18 can cause the algorithm to break the loop. While the cuts which are not yet generated may override the current node solution by changing it to a fractional solution, rendering it infeasible, generating another integer solution or changing its cost. Therefore, these uncertainties regarding the potential incumbent solutions need to be taken into account in order to maintain the global convergence.

To cope with this, we use two techniques. First, we make sure that the global upper bound is updated only when a reliable value is obtained, e.g., all SPs associated to a potential incumbent solution have been evaluated. Second, to overcome the uncertainties associated with the integer nodes for *binary* master variables, we make use of combinatorial cuts with the following form : $\sum_{i\in\{1,..,n\}:\bar{y}_i=0} y_i + \sum_{i\in\{1,..,n\}:\bar{y}_i=1}(1-y_i) \geq 1$, to forbid regeneration of the current integer solution $\bar{y}$. This cut eliminates the current solution by (1) making the current node infeasible, (2) generate another integer solution, or (3) leading to the generation of a new fractional solution. In the first case, the node can be pruned by the infeasibility rule. This does not affect the convergence because no other feasible solution can be extracted from that node. The second case is in fact a desirable situation as it may yield a better upper bound value. In the third case, the node will be added to the pool of active nodes.

Finally, for *general integer* master variables, the node uncertainty must be handled through synchronization. Thus, if the applied cuts do not affect the current integer solution, we need to wait until a violating cut associated with that solution is generated or all cuts have been applied.

6.5.2 Search techniques

In this part we propose some strategies to specify the scheduling and pool management decisions.

Solution and cut pool management

Considering the previously partially evaluated solutions and the new one at the current iteration, we need to decide which solution to choose and evaluate its associated (unevaluated) SPs. At each iteration, the master process broadcasts its solution to all slave processors. Each slave processor stores this solution in a pool and follows one of the following strategies to pick the appropriate one :

S1 : chooses solutions based on the *first-in-first-out* (FIFO) rule ;

S2 : chooses solutions based on the *last-in-first-out* (LIFO) rule ;

S3 : chooses solutions in the pool randomly. We experiment with two selection rules : (1) each solution in the pool has an equal chance to be selected, (2) each solution is assigned a weight of $\frac{1}{1+t}$, where t is number of the iterations since that specific solution has been generated, so that more recent solutions have higher chance to be selected.

Moreover, we use the *cut improvement* notion to identify the solutions which are no longer required to be evaluated, see Rei et al. (2009) for more information. Finally, we make use of the same techniques outlined in subsection 6.4.4 to manage the cut pool.

Solving SPs

We have implemented static work allocation because by equally distributing the scenario SPs, every process is almost equally loaded. Once the solution is chosen, we need to decide the order by which the associated SPs will be evaluated, because we may not evaluate all of them and it is important to give higher priority to those which tighten the MP formulation the most. The following strategies are considered :

SP1 : randomly choose the SPs ;

SP2 : assign a weight to each SP and then randomly select one based on the roulette wheel rule. The weights are set equal to the normalized demands for each SP.

SP3 : we observe that if a solution is infeasible, we may not need to solve all its SPs. This strategy first orders the SPs based on their demand sum and then assigns to each SP a *criticality* counter which increases by one each time that the SP is infeasible. Then, a SP with the highest criticality value is selected.

Solving the MP

This dimension specifies the waiting portion of the master processor before it re-optimizes the MP. We have proposed the following strategies :

MP1 : the master processor waits for at least $\gamma|\mathcal{S}|\%$ new cuts at each iteration ;

MP2 : the master processor waits for $\gamma|\mathcal{S}|\%$ of the cuts associated with the current solution ;

MP3 : this strategy is the same as the MP2 strategy, but with a mechanism to synchronize the processors according to the current state of the algorithm. In this regard, if the cuts added to the MP fail to affect the lower bound and/or regenerate the same solution, the MP waits until all the cuts associated with the current solution are delivered.

6.5.3 Cut aggregation

The asynchronous algorithm may only solve a subset of the SPs in each cluster. For this reason, cut aggregation as introduced in section 6.4.1 is not applicable in the context of our asynchronous parallel algorithm. To alleviate this issue, we define a recourse variable for each SP. Then, we add a cut of the form : $\sum_{s\in\hat{\mathcal{S}}_d}\rho_s\theta_s \geq \sum_{s\in\hat{\mathcal{S}}_d}\rho_s\Big(h_s - T_s y\Big)^\top \alpha_s$ for cluster $d \in \mathcal{D}$, where $\hat{\mathcal{S}}_d$ indicates the set of evaluated SPs in this cluster at the present iteration. Note that we could update this inequality in the following iterations when the remaining SPs in cluster d are evaluated. However, this process requires modifying the MP, which is not computationally efficient. We thus aggregate the cuts for the current solution separately from those associated with the previous iterations.

6.5.4 Partial information

The B&BC method is very sensitive to the feedback on its current solution. The asynchronous variant thus entails a larger number of iterations and a considerable amount of redundant work as it executes the next iteration with *partial information* on its current solution. In fact, the asynchronism may increase the computational cost of each iteration since the MP grows large at a faster pace. Moreover, the lower bound may progress slower since merely a subset of the cuts are applied to the MP at each iteration. The proof of convergence may as well be delayed due to the unavailability of the upper bound, especially when one wishes to solve the problem to a certain optimality gap. To overcome these drawbacks, we propose two classes of strategies acknowledging the fact that we solve a SP in order to : (i) calculate an upper bound and (ii) generate a cut.

The upper bound is used to assess the convergence and prune nodes. In our asynchronous method the delay in calculating the upper bound is usually due to long intervals between visiting two integer nodes, where we conventionally collect the feedbacks from the slave processors. Thus, by designing a module that can receive information from slave processors at every node of the search tree, we can overcome this issue and update the upper bound with minimal delay.

The generated Benders cuts are used to cutoff the subregion of the MP defined by the set of complicating variables. To address the issue related to having cuts merely for a subset of the recourse variables at each iteration, we have designed the following two strategies : (i) creating artificial SPs and (ii) propagating the generated cuts.

Creating artificial SPs

The MP does not wait for all SPs to be solved. Thus, having good cuts that can represent the unevaluated SPs is important to improve the efficiency of our asynchronous algorithm. To this end, we propose to create a set of artificial scenarios. The SP associated with each artificial scenario will then be solved to generate a valid cut to bound the recourse cost of the scenarios that remain unevaluated at the current iteration.

We assume that the recourse matrix and the costs are deterministic, i.e., $c_s = c$ and $W_s = W$, $\forall s \in \mathcal{S}$. We cluster the SPs into $|\mathcal{G}|$ groups according to the similarity measure. Note that the cardinality of clusters may not be equal. Then, to generate the artificial SP, $g \in \mathcal{G}$, we set $h_g = \sum_{s\in\mathcal{S}_g} \beta_s h_s$ and $T_g = \sum_{s\in\mathcal{S}_g} \beta_s T_s$, where $\mathcal{S}_g$ is the set of scenarios in cluster $g \in \mathcal{G}$ with $\mathcal{S} = \cup_{g\in\mathcal{G}}\mathcal{S}_g$, and β_s is the weight associated with scenario $s \in \mathcal{S}_g$ such that $\sum_{s\in\mathcal{S}_g} \beta_s = 1$.

Proposition 6.2. *Any extreme point α^g and extreme ray r^g of the artificial subproblem $g \in \mathcal{G}$ gives a valid optimality cut $\sum_{s\in\mathcal{S}_g} \beta_s\theta_s \geq (h_g - T_g y)^\top \alpha^g$ or a feasibility cut $0 \geq (h_g - T_g y)^\top r^g$.*

Proof. See Appendix N. □

An important issue in deriving the artificial cuts lies in setting the β weights. The following theorem suggests an optimal way to set these weights.

Theorem 6.1. *Maximum bound improvement by the artificial scenario $g \in \mathcal{G}$ is attained when the convex combination weight β_s for scenario $s \in \mathcal{S}_g$ is $\dfrac{\rho_s}{\sum_{s\in\mathcal{S}_g} \rho_s}$.*

Proof. See Appendix O. □

Note that we make use of the artificial scenarios to bound the recourse variable of those SPs which remain unevaluated at the present iteration. The following corollary suggests a strategy to further tighten the associated cuts.

Corollary 6.1. *Let $\bar{\mathcal{S}}_g$ be the set of evaluated SPs in cluster $g \in \mathcal{G}$ at the current iteration. Then solving the artificial SP using a smaller set $\mathcal{S}_g \backslash \bar{\mathcal{S}}_g$ yields a tighter cut for the remaining SPs. This follows the aggregation step in the proof of Proposition 6.2.*

From corollary 6.1, we observe that there is no need to apply the scenario creation strategy in the synchronized and sequential algorithms.

Cut propagation

In this part, we propose to propagate the generated cuts in order to derive additional approximate cuts for the set of unevaluated SPs. We make the same assumption on the problem's structure as in section 6.5.4.

Proposition 6.3. *Given an optimality cut* $\theta_s \geq (h_s - T_s y)^\top \alpha_s$ *associated with the SP* $s \in \mathcal{S}$*, we can generate a valid optimality cut* $\theta_{s'} \geq (h_{s'} - T_{s'} y)^\top \alpha_s$ *for SP* $s' \in \mathcal{S} : s' \neq s$ *without solving the SP for scenario* s'*.*

Proof. Based on our assumption, the dual SP (6.3) becomes $\max_{\alpha \in \mathbb{R}^l_+} \{(h_s - T_s \bar{y})^\top \alpha : W^\top \alpha \leq c\}$ for every $s \in \mathcal{S}$. Thus, all SPs have the same dual polyhedron for which each of its extreme points and rays gives a valid cut. □

It is not computationally viable to propagate any dual solution due to time consuming calculations and handling requirements. We thus generate propagated cuts only for those recourse variables whose associated SP remains unevaluated at current iteration. Also, if a dual value yields a cut which is not violated by the current solution, it will not be used for propagation. Finally, for SP s' we generate a propagated cut using the solution from SP s if the latter has the greatest *dominance* value over the former in the set of evaluated SPs at the current iteration. Note that the dominance value of SP s over SP s' is calculated as the total number of the elements $j \in \{1, ..., l\}$ for which $h_s^j \geq h_{s'}^j$ and ${T_s^j}^\top y \geq {T_{s'}^j}^\top y$ for every $y \in Y$ (Crainic et al., 2016).

6.5.5 The overall framework

The overall framework of the proposed asynchronous algorithm with its various components is depicted in Figure 6.2.

Each processor starts with an initialization step where a series of tasks are executed. At the master processor, the MP is created, scenarios are grouped (based on similarity criteria) and assigned to each slave processor, and the recourse variables are clustered for the cut aggregation purpose. Each slave processor waits to receive the assigned scenarios and then creates their actual mathematical formulation.

In the first phase, each time that the LP relaxation of MP is solved, the solution is broadcast to all slave processors. Then, the master processor waits to receive the feedback from these processors based on the chosen scheme from the *solving MP* strategies. In the next step, cuts from the pool are selected and aggregated using the appropriate strategy for *cut management*.

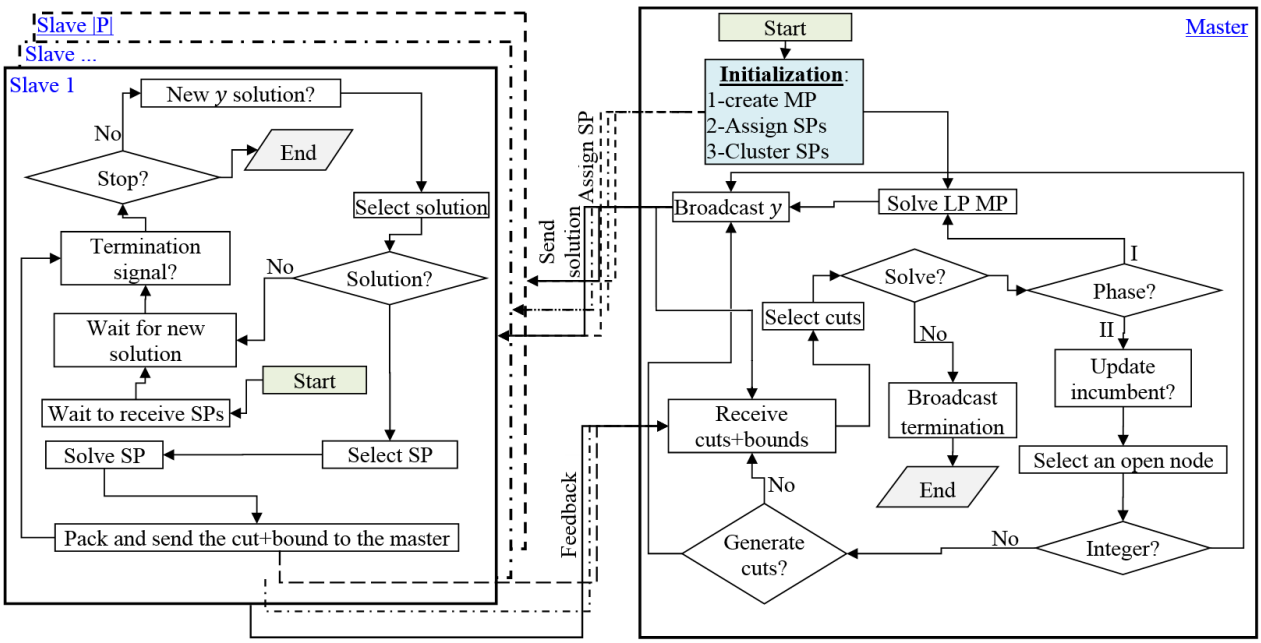

Figure 6.2 Flowchart of the asynchronous parallel method

If the first phase is not yet optimized, we repeat the same steps. Otherwise, we proceed with the branching phase.

In the second phase, we check if the incumbent solution can be updated before choosing and evaluating an open node. If the selected node is integer, we broadcast the solution to all processors. However, if the node solution is fractional, we need to decide if we want to generate cuts associated with that solution based on the cut generation strategies of section 6.8.1. Before evaluating the selected node, we check the cut pool to see if new cuts exist or not. If the selected node is fractional, we do not wait for cuts to be generated. For the integer nodes, however, we wait for feedback from slave processors depending on the *solving MP* strategy that we are using. Finally, if one of the stopping criteria is satisfied, we broadcast the termination signal to stop the algorithm.

All slave processors execute the same process but independently. At each step, they check if a new master solution is broadcast. If so, they receive the solution and store it in a pool. However, if there is no solution in the queue, they do not wait and proceed by choosing a solution from the pool based on the appropriate *solution management* strategy. If the solution pool is empty for any slave processor, it waits until a new one or the termination signal is broadcast. At next step, they select one of their local SPs based on the *solving SPs* strategy and evaluate it. Finally, each processor sends back to the master processors the generated

information (i.e., cut and bound). If no termination signal has been received, the process repeats.

6.6 Hybrid parallel Benders decomposition algorithm

We observed that when solving the MP and SPs is quick, re-optimizing the MP with partial feedback from the slave processors can yield efficiency drawbacks. This is particularly the case during the first phase of our asynchronous algorithm. For this reason, we hybridize the two previous parallelization strategies by solving the first phase with the synchronized and the second phase with the asynchronous strategies. We refer to this strategy as *hybrid parallelism*.

6.7 Implementation details

We solve all LP and MILP problems using IBM ILOG CPLEX 12.7. All programs are coded in C++ environment. The code is compiled with g++ 4.8.1 performed on Intel Xeon E7-8837 CPUs running at 2.67GHz with 64GB memory under a Linux operating system. The B&C algorithm was also implemented using CPLEX's callable libraries. We solve the extensive formulation with CPLEX's default setting, and turn off presolve features for our Benders-based algorithms.

To further accelerate the presented B&BC algorithms, we also incorporate some classical acceleration strategies. We have thus implemented the following techniques in all of our algorithms : (i) warm start strategy, (ii) valid inequalities for the MP, (iii) valid inequalities for each SP, and (iv) Pareto-optimal cuts. The complete details of these strategies can be found in Rahmaniani et al. (2017b). Finally, as local branching was shown to effectively accelerate the B&BC method (Rei et al., 2009), we turn on the local branching feature of CPLEX in the second phase of our B&BC algorithms.

6.7.1 Test instances

To test our method, we address the well-known *Multi-Commodity Capacitated Fixed-charge Network Design Problem* with *Stochastic Demand* (MCFNDSD). This problem naturally appears in many applications (Klibi et al., 2010) and it is notoriously hard to solve (Costa, 2005; Crainic et al., 2011). The complete detail of this problem is given in Appendix P. To conduct the numerical tests, we have used the **R** instances which are widely used in the literature, e.g., Chouman et al. (2017); Rahmaniani et al. (2017b); Crainic et al. (2016, 2011);

Boland et al. (2015). These instances have up to 64 scenarios. To generate a larger number of scenarios, we have followed a procedure similar to the one used by Boland et al. (2015). For the numerical assessment of the strategies, we have considered a subset of the instances : r04-r10 with correlation of 0.2 and cost/capacity ratio of 1, 3, or 9 which accounts for 21 instances. This subset of the **R** family corresponds to the instances most commonly tackled in the literature (Rahmaniani et al., 2017b; Crainic et al., 2016). In the following part of the computational experiments, where we study the performance of our method versus alternative methods, we have considered a larger number of instances, i.e., r04-r11 accounting for 200 instances. The description of these instances is given in Appendix Q.

6.7.2 Implementation of the asynchronous algorithm

We discussed various search strategies whose combination gives a very large number of algorithms to test. Presenting the numerical results for all of them is clearly beyond the length of this article. For this reason, we provide some insights for those strategies which we do not intent to present numerical results for.

With respect to *solution management* strategies, we observed that LIFO outperforms both FIFO and random strategies. The main reason is that Benders method, as “a dual algorithm", is very sensitive to the feedback on its current solution. In the both FIFO and random selection strategies, “older" solutions are usually selected. As a result, the generated cuts are dominated or they have a very limited impact on the MP. Thus, the MP generates a solution which is not very different (in terms of quality) from the previous iteration(s) and the lower bound progresses very slowly. Furthermore, in both strategies the upper bound improves at a slower pace compared to LIFO, although they might update the upper bound more frequently in the initial iterations. This is because the FIFO and random selection strategies need a much larger number of iterations to actually find a high quality feasible solution that gives a tight bound. As a result, we henceforth only consider the LIFO as solution selection strategy.

With respect to *solving SPs*, we realized that, on average, the random selection of the SPs performs better than other strategies. This is because the random selection of the SPs ensures the diversity of the cuts applied to the MP. In ordering based strategies, after a certain number of iterations, a specific set of the recourse variables is bounded at each iteration. Hence, the lower bound progresses slowly and the algorithm performs poorly. Finally, the random selection based on the criticality weights tends to perform better than pure random selection. This is because higher priority is given to the indicator SPs and the diversity of the cuts applied to the MP is maintained. Therefore, we consider this strategy to select SPs

for evaluation.

Finally, we consider the proposed strategies in the *solving MP* dimension. We have decided to use both MP2 and MP3. This is because of two reasons. First, we observed that applying cuts associated with the current solution is necessary. Otherwise, the algorithm performs poorly due to the same reasons pertinent to the FIFO strategy. Second, with adoptive synchronization points the late convergence proof in the first phase of the algorithm will be alleviated significantly. On the contrary, synchronization causes overheads during the second phase of the algorithm because master processor can proceed with evaluating the open nodes of the branch-and-bound tree while the cuts are being generated. Therefore, we use MP3 in the first and MP2 in the second phase of our algorithm.

6.7.3 Stopping criteria and search parameters

In solving each stochastic instance, we have set the stopping optimality gap at 1%. The total time limit is set at 2 hours. To solve the LP relaxation of the problem at the root node, we have considered half of the maximum running time limit. The parallel variants are run on 5 processors unless otherwise specified. In the cut generation strategy, τ, λ, and $\hat{\lambda}$ values are set to 0.5%, 10^{-1}, and 10^{-2} . The set of scenarios in the feasibility repair problem are chosen based on their demand sum. We have sorted the scenarios based on their total demand and chosen the first 5 scenarios.

6.8 Computational results

In this section, we present the numerical assessments of the proposed parallelization strategies. We first study different versions of our parallel B&BC algorithms to evaluate the limitations and impact of the proposed acceleration techniques. The second part of the analysis is devoted to test the speedup and scalability of our parallel algorithms. Finally, we conduct a comparison between our exact algorithms and CPLEX to benchmark their performance.

6.8.1 Synchronized parallel algorithm

In this section, we investigate the impact of the proposed acceleration strategies, namely the cut management, cut aggregation and cut generation in context of the synchronous parallel B&BC algorithm. Note that we have activated the proposed strategies one by one to observe their cumulative additive impact over the performance. Thus, the basic algorithm in each subsection is the best variant obtained from the previous subsection.

Cut management

We first study the impact of the cut management strategy. We compare the method proposed in section 6.4.4 to that commonly used in the literature, where cuts with high slack value are removed, see, e.g., Pacqueau et al. (2012). In Table 6.2, the average running time in seconds, optimality gap in percentages, and number of removed cuts, shown with #Cut, are reported for each cut management strategy.

Table 6.2 Numerical results for various cut management strategies

	NoCutManagement		**SlackCutManagement**			**DominanceCutManagement**		
	Time(ss)	Gap(%)	Time(ss)	Gap(%)	#Cut	Time(ss)	Gap(%)	#Cut
r04	2422.60	1.00	2424.57	1.26	151.00	1802.97	0.36	1616.00
r05	1133.64	0.47	517.22	0.47	164.33	500.64	0.47	942.67
r06	3270.45	1.76	2656.59	2.06	602.33	2601.43	2.12	22.00
r07	2437.16	3.32	2439.03	2.37	13.67	2141.70	2.06	1599.33
r08	2534.89	3.36	2513.26	3.56	50.67	2315.18	2.40	292.00
r09	4898.53	4.01	4737.94	4.24	163.67	4780.70	4.25	868.67
r10	4977.60	7.77	4927.27	5.25	645.00	4931.94	6.23	108.00
Ave.	**3096.41**	**3.10**	**2887.98**	**2.75**	**255.81**	**2724.94**	**2.56**	**778.38**

It is important to note that none of the methods deteriorates the LP bound at the root node. We observe from Table 6.2 that cleaning up the useless cuts yields positive impact on the performance. The slack based notion keeps many of the useless cuts in the MP while the proposed method removes a much larger number of them. Thus, for small and medium instances, we observe a clear advantage of the proposed method. However, for larger instances, the impact of removing cuts on running is less significant because the algorithm reaches the time limit. We here note that most of the removed cuts are associated with early iterations. Finally, we observe that there is no direct relation between the number of the cuts removed and the performance of the B&BC method. This is because of the heuristic nature of both strategies.

Cut aggregation

We ran the algorithm with 11 different cluster sizes in order to study the impact of various cut aggregation levels on the performance. The comparative results in terms of total running time are depicted in Figure 6.3. Note that the value on each column gives the average optimality gap in percentages.

We observe from Figure 6.3 that neither of the two conventional strategies, i.e., single cut (column labeled “1") and multi-cut (column labeled “1000"), gives the best results. Although the latter performs noticeably better than the former. The best aggregation level is associated

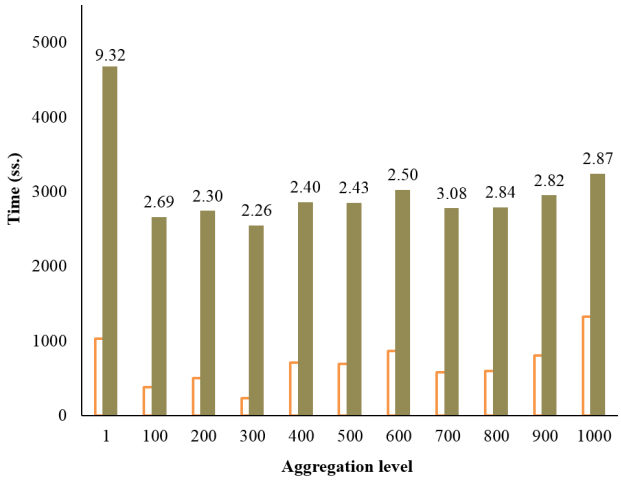

Figure 6.3 Comparison of various cut aggregation levels for the synchronized parallel algorithm

with a cluster size of 300. All aggregation levels are able to solve 66.67% of the instances within the 2 hours time limit, except for the single cut method which could only solve 61.90% of the instances. In addition, we observe that the difference in running time among some aggregation levels is small. This is because the algorithm reaches the maximum run time limit for 33.33% of the instances. Thus, if we only consider the instances that are solved by all aggregation levels, we observe more significant differences in the running times. These results are presented by the empty half bars in Figure 6.3.

Comparing the results to those of the sequential algorithm, see Figure R.1 in Appendix R, we observe that a larger cluster size gives the best results for the sequential algorithm. This is because the sequential algorithm performs less iterations in the same amount of time and thus, it is more sensitive to the loss of information in the aggregation step.

Cut generation strategy

We next study the cut generation strategies of section 6.4.3. For computational purposes, we have set Γ to 10, 100 and ∞ and compare the results to the case for which we generate cuts merely at the root node, i.e., $\Gamma = 0$. For each Γ value, we have reported numerical results for two cut generation strategies : “Single" indicates generating cuts once per node and “Multi" indicates generating cuts for each node as long as the lower bound improves by more than 0.5%. Also, we have studied two cases for each strategy : (i) the fractional solutions are directly used and (ii) the feasibility of the infeasible ones is restored. The numerical results are summarized in Table 6.3.

Table 6.3 Numerical results for various cut generation strategies

Γ	Strategy	NoFeasibilityRestoration			FeasibilityRestoration		
		Time(ss)	Gap(%)	Sol.(%)	Time(ss)	Gap(%)	Sol.(%)
∞	Single	2737.66	20.19	57.14	2694.15	10.95	66.67
∞	Multi	3045.31	29.38	57.14	3164.12	24.80	61.90
100	Single	2538.98	16.27	52.38	2611.14	10.58	66.67
100	Multi	2999.89	25.00	57.14	3050.19	24.84	66.67
10	Single	**2398.36**	**2.16**	**71.43**	2405.82	2.41	71.43
10	Multi	2612.17	2.48	71.43	2583.14	2.40	71.43
0	-	2549.10	2.26	66.67	-	-	-

Here “Sol.(%)" gives percentage of the solved instances. From Table 6.3 we observe that generating cuts for many fractional nodes considerably increases the optimality gap. This happens because of two main reasons. First, generating cuts for many nodes is computationally expensive and second, it causes considerable handling costs. For the same reasons, the “Multi" strategy underperforms the one with the “Single" module. On the contrary, generating cuts merely for the first 10 nodes performs better or equal to generating cuts only at the root node, such that the average running time and optimality gap reduce by 150.74 seconds and 0.1%, while the percentage of solved instances increases by 4.76%.

We next examine the strategies proposed to round the fractional solutions. The first strategy, denoted *Round*, rounds each fractional value to its closest integer value. The second strategy, denoted *Upward*, rounds each fractional value greater and equal than 0.1 to 1 and 0 otherwise. The third strategy employs a restricted MIP to find a close integer feasible solution to the current fractional solution. The numerical results are summarized in Table 6.4.

Table 6.4 Numerical results for the rounding based cut generation strategies

Strategy	NoFeasibilityRestoration			FeasibilityRestoration		
	Time(ss)	Gap(%)	Sol.(%)	Time(ss)	Gap(%)	Sol.(%)
MIP	-	-	-	**2486.78**	**2.31**	**71.43**
Round	2629.98	2.20	61.90	2591.03	2.25	61.90
Upward	2579.79	2.35	66.67	2606.85	2.44	71.43
$\Gamma = 0$	2549.10	2.26	66.67	-	-	-

Note that, following the observations in Table 6.3, we have applied these strategies only at the first 10 fractional nodes. Comparing Tables 6.3 and 6.4, we observe that the rounding strategies do not perform better than directly using the fractional solutions. The main reason is that the cuts associated to the rounded solutions do not, in general, affect the local bounds. In addition, CPLEX’s default heuristics and local branching are active in our implementations. This makes the marginal impact of the rounding strategies negligible.

6.8.2 Asynchronous parallel algorithm

In this part, we study the proposed strategies for the asynchronous parallel algorithm. We first study different synchronization levels as controlled by the γ parameter. Then, we analyze the proposed acceleration techniques.

The synchronization level

Figure 6.4 depicts the impact of the γ value on the running time. The values on this figure represent the average optimality gaps in percentage.

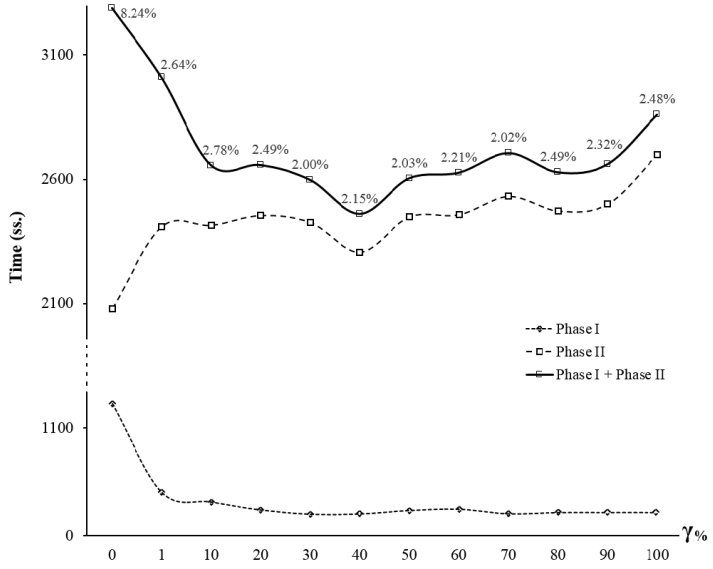

Figure 6.4 Effect of waiting portion of the master processor (gamma value) on the performance

We observe that the best performance lies neither at 100% nor at 1%. It is always numerically better to wait for some percentage in between. Moreover, comparing Figures 6.4 and R.1, we observe that full asynchronism (i.e., $\gamma = 0$) underperforms even the sequential algorithm. The best result is attained when the master processor waits until 40% of the SPs associated with the current solution are solved. This synchronization level solves 66.67% of the instances with the average optimality gap of 2.15% in 2465.26 seconds. This is comparable to the synchronous algorithm without the cut generation strategies, see Figure 6.3.

We observe that waiting for a larger portion of the SPs is more desirable when solving the LP relaxation of the MP, while the contrary is true in the second phase of the algorithm. In the former case, we gain nothing from quickly resolving the MP since no other useful task can be executed. In the latter case, however, we can perform useful work (i.e., evaluating open

nodes of the search tree) while the cuts are being generated. We thus consider $\gamma = 100\%$ and 0% in the first and second phase of our hybrid algorithm.

Scenario creation

In Table 6.5 we investigate the impact of the artificial SPs on the convergence of our asynchronous algorithm.

Table 6.5 Impact of the artificial subproblems on performance of the asynchronous algorithm

	NoArtificialSP			NumberOfArtificialSPs=$\|\mathcal{P}\|$		
	Time(ss.)	Gap(%)	Sol.(%)	Time(ss.)	Gap(%)	Sol.(%)
r04	689.50	0.75	100.00	759.39	0.73	100.00
r05	274.84	0.51	100.00	286.64	0.42	100.00
r06	2518.45	1.86	66.67	2546.88	1.48	66.67
r07	1442.27	0.84	66.67	1412.31	0.90	66.67
r08	2521.60	2.65	66.67	2466.25	2.02	66.67
r09	4898.42	3.98	33.33	4859.83	3.46	66.67
r10	4911.77	5.07	33.33	4880.33	4.53	33.33
Ave.	2465.26	2.25	66.67	2458.80	1.93	71.43

We observe in Table 6.5 that the creation of artificial scenarios increases the percentage of the solved instances by 4.76% while the average time is almost unchanged. The running time with artificial SPs increases for small instances and for larger instances the time improvement is not very much noticeable. In small problems additional time is spent on generating cuts while they could have been solved quicker. For larger instances, in many cases the algorithm reaches the time limit and thus the improvement on the average time does not show. We observe, however, that the average optimality gap has been reduced. Thus, the artificial SPs are a valid strategy to accelerate the asynchronous algorithm.

Cut aggregation

Figure 6.5 reports the impact of the proposed cut aggregation scheme for different cluster sizes. The value on each bar shows the average optimality gap in percentages.

We observe that clustering improves the convergence of the algorithm. The best aggregation level is reached at the size of 500, which is larger than the best value found for the synchronized algorithm, i.e., a cluster size of 300. With this aggregation level, our asynchronous algorithm solves 76.19% of the instances with an optimality gap of 1.46% in 2219.42 seconds, which outperforms the synchronized algorithm. Moreover, the single cut method does not perform much differently from other aggregation levels. This is because of introducing a recourse variable for each SP in the aggregated cuts.

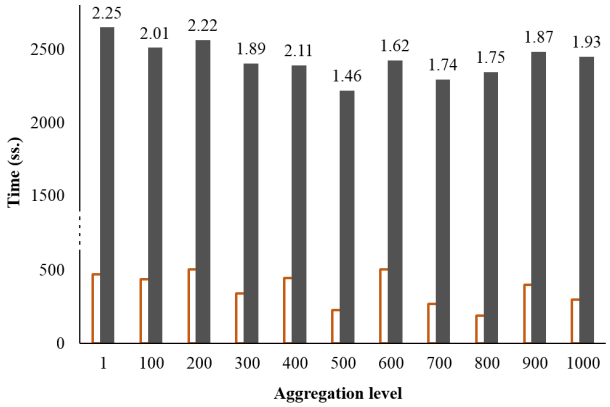

Figure 6.5 Comparison of different cut aggregation levels in context of the asynchronous method

Cut propagation

To assess the usefulness of the proposed cut propagation strategy, we compare our algorithm with and without the cut propagation module. Table 6.6 summarizes the results in terms of average time requirement and number of iterations. Note that the cut propagation is performed merely at the first phase of the algorithm. We have thus reported the time required and number of iterations to find the optimal LP solution.

Table 6.6 Impact of the cut propagation on the LP phase of the asynchronous algorithm

	WithoutPropagation		WithPropagation	
	Time(ss.)	#Iter.	Time(ss.)	#Iter.
r04	29.62	17.00	111.24	13.67
r05	50.88	21.33	213.86	14.00
r06	307.41	32.67	1377.90	18.67
r07	19.98	15.67	286.50	14.00
r08	59.61	21.00	415.22	15.33
r09	546.43	30.67	1085.35	16.00
r10	1158.31	41.67	1591.48	23.00
Ave.	310.32	25.71	725.94	16.38

We observe from Table 6.6 that the cut propagation reduces number of the major iterations. However, it increases the time requirement to optimize the LP relaxation of the problem. This is partially due to the additional time we need to spend on handling the propagated cuts and checking their violation. In addition, we can solve the LP relaxation of the considered instances within the considered time limit and the cut propagation does not further tighten this bound. We thus believe that the proposed cut propagation strategy can be very useful for problems with a fairly small number of hard-to-solve SPs rather than many of easy-to-solve SPs. Since this is not the case in our problem, we do not include this strategy in our method.

6.8.3 Speedup of the parallel algorithms

In Figure 6.6, we compare our parallel algorithms on 2, 3, 5, 10, 15 and 20 processors. In this experiment, we have considered only those instances that the sequential algorithm could solve within a 10-hour limit in order to have a clear sense of the speedups. Note that the value on each bar indicates the speedup ratio calculated as Sequential time/Parallel time.

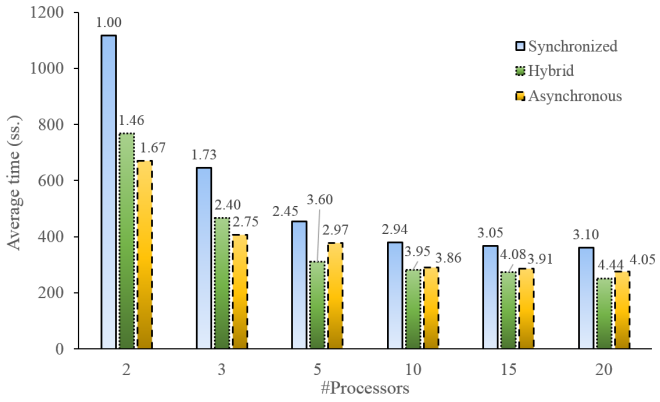

Figure 6.6 Speedup rates of our parallel Benders decomposition algorithms

The first interesting observation is related to the use of 2 processors, i.e., 1 slave and 1 master processor. In this case, the synchronized algorithm performs worse than the sequential algorithm because of the cut generation which slows down the algorithm. Our asynchronous algorithm, on the contrary, performs much better. This is clearly due to the better use of the processors in our asynchronous method since the two processors are less dependent. Also, we observe that the hybrid algorithm is better than the synchronized one, although they are roughly the same during the first phase of the algorithm. Thus, the speedup rate in our hybrid algorithm must be due to its asynchronous part. This indicates that our asynchronous method has noticeably reduced the computational bottleneck at the master level.

The speedups do not increase monotonically with the number of the processors. This is because increasing the number of slave processors does not alleviate the bottleneck at the master processor, although it significantly accelerates the cut generation cycle. We observe that the synchronous algorithm reaches super linear speedup during the LP phase of the algorithm, while it fails to reach even linear speedup overall. This is due to the second phase of the algorithm where the slave processors are not efficiently used and the heavy work is carried out by a single processor, i.e., master. The same situation also applies to the asynchronous and hybrid algorithms.

Our third observation concerns the hybrid method. It outperforms the synchronized method

because of using non-blocking communications in the branching phase, which is the most time consuming part of the algorithm ; see Figure 6.4. The advantage of the hybrid algorithm over the asynchronous method becomes more evident for larger instances in which solving the LP relaxation is noticeably time consuming. Moreover, we observe that its efficiency exceeds or becomes closer to the asynchronous method as the number of processors increases. This justifies the development of the hybrid algorithm.

6.8.4 Comparison with CPLEX

In this part, we compare our asynchronous and hybrid algorithms versus the latest version of CPLEX for all instances, see Table Q.1 in Appendix Q. In doing so, we ran our algorithms until reaching the same optimality gap which is obtained by CPLEX after 10 hours. Note that all algorithms are run on 15 processors. The average speedup rates are reported in Figure 6.7. These values are rounded to the closest integer point and they are obtained from dividing CPLEX's time requirement to that of our methods.

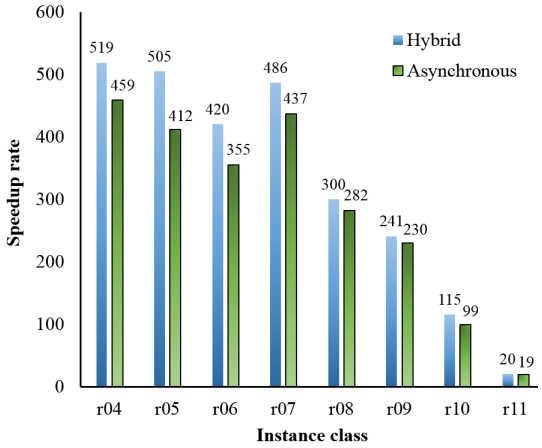

Figure 6.7 The speedup rate of our asynchronous and hybrid parallel algorithm compared to CPLEX

The asynchronous and hybrid parallel algorithms are, on average, 286.48 and 325.69 times faster than CPLEX in finding a solution of equal or better quality. We observe that the speedup for larger instances is smaller because solving their LP relaxation takes up to one hour. This makes the speedup rates in scale of 10. Moreover, the hybrid parallelization reaches a better speedup rates since it solves the LP relaxation faster.

To complete the comparative study in this section, we compare the optimality gaps obtained by our parallel algorithms to that of CPLEX for a run time limit of 2 hours. Note that if

CPLEX fails to find a feasible solution, we have set the optimality gap at 100%. The average optimality gap for each instance class is summarized in Table 6.7.

Table 6.7 The average optimality gap obtained by each method after 2 hours

	CPLEX	**Asynchronous**	**Hybrid**
r04	14.49	0.33	0.34
r05	31.84	0.48	0.46
r06	56.11	2.00	2.75
r07	33.73	0.34	0.50
r08	43.90	2.18	2.48
r09	58.07	3.25	2.72
r10	56.15	3.58	2.28
r11	86.86	7.65	6.82
Ave.	**47.64**	**2.48**	**2.29**

We observe from Table 6.7 that CPLEX fails to handle even small instances of the considered stochastic problem. Furthermore, the hybrid algorithm reaches better optimality gaps than the asynchronous method for larger instances.

6.9 Conclusions and remarks

We studied parallelization strategies for the Benders decomposition method in which the subproblems are concurrently solved on different processors and the master problem on a single processor. We implemented the algorithm in a B&C framework and presented synchronous, asynchronous and hybrid parallelization frameworks along with various acceleration strategies.

Reporting numerical results on hard benchmark instances from stochastic network design problems with 1000 scenarios, we observed (super-)linear speedups when the master problem does not computationally dominate the algorithm. This is particularly true when solving the LP relaxation of the master problem. In similar cases, the synchronized method performed generally better than the asynchronous algorithm. On the larger instances, our parallel algorithms did not reach a linear speedup. Moreover, we observed that our parallelizations did not scale with the number of the processors. The main reason for this is the fact that the most significant computational bottleneck of the parallel algorithm is solving the master problem sequentially on a single processor. Also, the proposed asynchronous and hybrid algorithms displayed a better performance compared to the synchronous method. Compared to CPLEX, they were more than 286 times faster to obtain the same optimality gap. In addition, for the same time limit, they could also obtain optimality gaps that were more than 19 times lower than CPLEX.

This research opens the way for a number of interesting issues to be considered in future works. First and foremost, the master problem needs to be solved in parallel in order to reach a scalable algorithm. Second, it is worthwhile to study the proposed cut propagation strategy for problems in which generating a single optimality cut is significantly time consuming. Third, some of the well-known acceleration strategies for the Benders method need to be revisited in order to be properly applied in the parallel environment. An example would be the partial decomposition strategy of Crainic et al. (2016). Last but not least, heuristics are widely used to accelerate the BD method, but we are not aware of any integration of these methods in a parallel (cooperative) framework.

CHAPTER 7 GENERAL DISCUSSION

In this dissertation, we studied decomposition-based solution methods for combinatorial optimization problems with particular interest in stochastic integer programming. We strove to improve the efficiency of the Benders decomposition algorithm and also, proposed a novel decomposition method. We tested our developments on benchmark instances from *multi-commodity capacitated network design*, *capacitated facility location*, and *network interdiction* problems.

In the first essay of this dissertation, we reviewed and synthesized decades of research on the Benders decomposition method for the first time in the literature. We discussed the classical algorithm, the impact of the problem formulation on its convergence, and its relationship to other decomposition methods. We introduced a taxonomy of algorithmic enhancements and acceleration strategies based on the main components of the algorithm. The taxonomy provided a framework to synthesize the literature and to identify shortcomings, trends, and potential research directions. We also discussed the use of the Benders decomposition to develop efficient (meta-)heuristics, described the limitations of the classical algorithm, and presented extensions enabling its application to a broader range of problems.

Following our extensive literature review, in the second essay of this dissertation, we addressed some of the most important drawbacks of the Benders decomposition algorithm. We proposed various acceleration techniques, including the use of cutting planes (one for the master problem and one for each subproblem), partial decomposition, heuristics, stronger cuts, reduction and warm-start strategies. This yielded an effective Benders decomposition algorithm capable of efficiently solving challenging instances of the well-known multi-commodity capacitated network design problem with demand uncertainty. The numerical results confirmed the clear advantage of the proposed accelerations strategies and the resulting algorithm over the existing ones.

In the third essay, we developed a new decomposition method which combines the complementary advantages of the classical decomposition approaches, i.e., Benders and Lagrangian dual decomposition methods. The development of our method was based on a specific reformulation of the Benders subproblem, where local copies of the master variables are introduced and then priced out into the objective function. By imposing the integrality requirement on the copied variables, we showed that at the fractional points of the master problem, this method generates stronger optimality and feasibility cuts than those of the Benders decomposition method. In fact, one can obtain an optimal integer solution at the root node of the

Benders master problem by using these cuts. In addition, we observed that this method is capable of producing high-quality primal bounds at the initial iterations of the algorithm and it can be applied to a wider range of the optimization problems than the classical Benders decomposition method. We used our method to solve various combinatorial optimization problems and compared the results to those obtained by the classical decomposition methods as well as CPLEX 12.7. The obtained results were highly encouraging.

Finally, we studied parallelization strategies for the Benders decomposition method in the fourth essay. We focused on the strategies where the subproblems are solved in parallel and the master problem is solved sequentially. Unlike the existing works, we developed our parallel algorithms in a branch-and-cut framework. We argued that this method is different from the existing parallel branch-and-cut methods. Moreover, we designed a framework that allows relaxing the synchronization requirements among the master and slave processors. To increase the efficiency of the algorithms, we proposed various acceleration strategies. We conducted an extensive numerical study on benchmark instances from stochastic network design problems. The results indicated that our asynchronous algorithm can reach higher speedup rates compared to the conventional parallel methods. We also observed that it is several orders of magnitude faster than the state-of-the-art solvers.

CHAPTER 8 CONCLUSION AND RECOMMENDATIONS

In short, this dissertation provided various numerical and theoretical contributions to two main branches of mathematical optimization, i.e., mixed-integer programming and stochastic programming, by developing new algorithms, proposing various techniques to accelerate the classical algorithms, and solving many challenging benchmark instances. In addition, this doctoral research opens the way for a number of interesting issues to be considered in future works. This constitutes the main subject of the this Chapter.

In the first essay, we reviewed the vast literature associated with the Benders decomposition algorithm. However, our main focus was on the acceleration techniques proposed for the classical algorithm. In particular, there are various extensions of this algorithm and it has inspired the development of some new algorithms, see, e.g., sections 2.8 and 2.9 of this dissertation. Furthermore, the associated literature is growing very fast. For instance, since the publication of our literature review more than 1940 scientific works have mentioned "Benders Decomposition" according to GoogleScholar. Therefore, this work can be extended by presenting complementary literature reviews and updating our survey.

In the second and fourth essays, we observed that solving the master problem is a major hurdle in handling large-scale mixed-integer (stochastic) problems. For instance, it prevented us from having a scalable parallel Benders decompositions algorithm. Also, although it starts with a fairly tight upper bound at the root node, it rarely improves this bound. Thus, developing advanced acceleration strategies to parallelize and better manage the branch-and-bound tree appear important subjects for the future research. In the same line of research, we encourage the development of a parallel algorithm in which our asynchronous Benders decomposition method and a primal heuristic are integrated into a cooperative framework.

The algorithm presented in the third essay was in its general form and there are many possibilities to further improve it. In the proposed algorithm each subproblem is associated with a single scenario while having subproblems consisting of multiple scenarios can yield tighter lower bounds and better incumbent solutions. We thus encourage the development of clustering strategies in the context of the proposed method. In addition, many researchers have focused on improving both Benders and Lagrangian decomposition methods. These strategies can be adopted and incorporated into our method to further enhance its numerical performance. Last but not least, the proposed method guarantees convergence even when the Benders subproblems are nonlinear and/or mixed-integer programs. Thus, the numerical assessment of this method versus the classical algorithms, such as integer L-shaped and outer

approximation methods, is certainly worthwhile.

REFERENCES

Aardal, Karen, Torbjörn Larsson. 1990. A Benders decomposition based heuristic for the hierarchical production planning problem. *European Journal of Operational Research* **45**(1) 4–14.

Achterberg, Tobias, Thorsten Koch, Alexander Martin. 2005. Branching rules revisited. *Operations Research Letters* **33**(1) 42 – 54.

Adulyasak, Yossiri, Jean-François Cordeau, Raf Jans. 2015. Benders decomposition for production routing under demand uncertainty. *Operations Research* **63**(4) 851–867.

Ahmed, Shabbir. 2010. Two-stage stochastic integer programming : A brief introduction. James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, J. Cole Smith, eds., *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, 1–10.

Ahmed, Shabbir. 2013. A scenario decomposition algorithm for 0–1 stochastic programs. *Operations Research Letters* **41**(6) 565 – 569.

Alonso-Ayuso, Antonio, Laureano F. Escudero, M. Teresa Ortuño. 2003. Bfc, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0–1 programs. *European Journal of Operational Research* **151**(3) 503 – 519.

Angulo, Gustavo, Shabbir Ahmed, Santanu S. Dey. 2016. Improving the integer L-shaped method. *INFORMS Journal on Computing* **28**(3) 483–499.

Archibald, T W, C S Buchanan, K I M McKinnon, L C Thomas. 1999. Nested benders decomposition and dynamic programming for reservoir optimisation. *Journal of the Operational Research Society* **50**(5) 468–479.

Ariyawansa, K. A., D. D. Hudson. 1991. Performance of a benchmark parallel implementation of the Van Slyke and Wets algorithm for two-stage stochastic programs on the sequent/balance. *Concurrency : Practice and Experience* **3**(2) 109–128.

Arthur, David, Sergei Vassilvitskii. 2007. k-means++ : The advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1027–1035.

Balas, Egon, Sebastián Ceria, Gérard Cornuéjols. 1993. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming* **58**(1) 295–324.

Barahona, Francisco, Ranga Anbil. 2000. The volume algorithm : producing primal solutions with a subgradient method. *Mathematical Programming* **87**(3) 385–399.

Bazaraa, Mokhtar S, Hanif D Sherali, Chitharanjan M Shetty. 2013. *Nonlinear Programming : Theory and Algorithms*. John Wiley & Sons.

Beale, E. M. L. 1965. Survey of integer programming. *Journal of the Operational Research Society* **16**(2) 219–228.

Beck, J. Christopher. 2010. Checking-up on branch-and-check. David Cohen, ed., *Principles and Practice of Constraint Programming – CP 2010 : 16th International Conference*, vol. 6308. Springer, Berlin, Heidelberg, 84–98.

Behnamian, J. 2014. Decomposition based hybrid VNS-TS algorithm for distributed parallel factories scheduling with virtual corporation. *Computers & Operations Research* **52** 181–191.

Ben-Tal, Aharon, Laurent El Ghaoui, Arkadi Nemirovski. 2009. *Robust optimization*. Princeton University Press.

Benders, Jacques F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4**(1) 238–252.

Benoist, Thierry, Etienne Gaudin, Benoît Rottembourg. 2002. Constraint programming contribution to Benders decomposition : A case study. *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*. CP '02, Springer-Verlag, London, UK, 603–617.

Birge, John R. 1985. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research* **33**(5) 989–1007.

Birge, John R, Francois Louveaux. 1997. *Introduction to Stochastic Programming*. Springer, New York.

Birge, John R, Francois V Louveaux. 1988. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research* **34**(3) 384–392.

Bloom, Jeremy A. 1983. Solving an electricity generating capacity expansion planning problem by generalized Benders' decomposition. *Operations Research* **31**(1) 84–100.

Bodur, Merve, Sanjeeb Dash, Oktay Günlük, James Luedtke. 2017. Strengthened Benders cuts for stochastic integer programs with continuous recourse. *INFORMS Journal on Computing* **29**(1) 77–91.

Boland, Natashia, Matteo Fischetti, Michele Monaci, Martin Savelsbergh. 2015. Proximity Benders : a decomposition heuristic for stochastic programs. *Journal of Heuristics* 1–18.

Boschetti, Marco, Vittorio Maniezzo. 2009. Benders decomposition, lagrangean relaxation and metaheuristic design. *Journal of Heuristics* **15**(3) 283–312.

Botton, Quentin, Bernard Fortz, Luis Gouveia, Michael Poss. 2013. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing* **25**(1) 13–26.

Brandes, K. T. 2011. Implementierung und analyse verschiedener strategien zur aggregation und disaggregation von multi-cuts im Benders dekompositionsverfahren. Master's thesis, Universität Paderborn, North Rhine-Westphalia, Germany.

Cai, Ximing, Daene C. McKinney, Leon S. Lasdon, David W. Watkins. 2001. Solving large nonconvex water resources management models using generalized Benders decomposition. *Operations Research* **49**(2) 235–245.

Canto, Salvador Perez. 2008. Application of Benders decomposition to power plant preventive maintenance scheduling. *European Journal of Operational Research* **184**(2) 759–777.

Carøe, Claus C, Jørgen Tind. 1998. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming* **83**(1-3) 451–464.

Carøe, Claus C., Rüdiger Schultz. 1999. Dual decomposition in stochastic integer programming. *Operations Research Letters* **24**(1) 37 – 45.

Cerisola, Santiago, Álvaro Baíllo, José M. Fernández-López, Andrés Ramos, Ralf Gollmer. 2009. Stochastic power generation unit commitment in electricity markets : A novel formulation and a comparison of solution methods. *Operations Research* **57**(1) 32–46.

Chaieb, Marouene, Jaber Jemai, Khaled Mellouli. 2015. A hierarchical decomposition framework for modeling combinatorial optimization problems. *Procedia Computer Science* **60**(Supplement C) 478 – 487.

Charnes, Abraham, William W Cooper. 1959. Chance-constrained programming. *Management science* **6**(1) 73–79.

Chen, Binyuan, Simge Küçükyavuz, Suvrajeet Sen. 2011. Finite disjunctive programming characterizations for general mixed-integer linear programs. *Operations Research* **59**(1) 202–210.

Chermakani, Deepak Ponvel. 2015. Optimal aggregation of blocks into subproblems in linear programs with block-diagonal-structure. *arXiv, Available at https ://arxiv.org/ftp/arxiv/papers/1507/1507.05753.pdf* .

Chouman, Mervat, Teodor Gabriel Crainic, Bernard Gendron. 2014. The impact of filtering in a branch-and-cut algorithm for multicommodity capacitated fixed charge network design. Publication CIRRELT-2014-35, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

Chouman, Mervat, Teodor Gabriel Crainic, Bernard Gendron. 2017. Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science* **51**(2) 650–667.

Codato, Gianni, Matteo Fischetti. 2006. Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research* **54**(4) 756–766.

Conforti, Michele, Gérard Cornuéjols, Giacomo Zambelli. 2010. Extended formulations in combinatorial optimization. *4OR* **8**(1) 1–48.

Contreras, Ivan, Jean-François Cordeau, Gilbert Laporte. 2011. Benders decomposition for large-scale uncapacitated hub location. *Operations Research* **59**(6) 1477–1490.

Cordeau, Jean-François, Federico Pasin, Marius M Solomon. 2006. An integrated model for logistics network design. *Annals of Operations Research* **144**(1) 59–82.

Cordeau, Jean-François, François Soumis, Jacques Desrosiers. 2001a. Simultaneous assignment of locomotives and cars to passenger trains. *Operations Research* **49**(4) 531–548.

Cordeau, Jean-François, Goran Stojković, François Soumis, Jacques Desrosiers. 2001b. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science* **35**(4) 375–388.

Corréa, Ayoub Insa, André Langevin, Louis-Martin Rousseau. 2007. Scheduling and routing of automated guided vehicles : A hybrid approach. *Computers & Operations Research* **34**(6) 1688–1707.

Costa, Alysson M. 2005. A survey on Benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research* **32**(6) 1429–1450.

Costa, Alysson M, Jean-François Cordeau, Bernard Gendron. 2009. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications* **42**(3) 371–392.

Costa, Alysson M, Jean-François Cordeau, Bernard Gendron, Gilbert Laporte. 2012. Accelerating Benders decomposition with heuristic master problem solutions. *Pesquisa Operacional* **32**(1) 03–20.

Côté, Gilles, Michael A. Laughton. 1984. Large-scale mixed integer programming : Benders-type heuristics. *European Journal of Operational Research* **16**(3) 327–333.

Côté, Jean-François, Mauro Dell'Amico, Manuel Iori. 2014. Combinatorial Benders' cuts for the strip packing problem. *Operations Research* **62**(3) 643–661.

Crainic, Teodor Gabriel. 2000. Service network design in freight transportation. *European Journal of Operational Research* **122**(2) 272 – 288.

Crainic, Teodor Gabriel. 2015. Parallel meta-heuristic search. Publication CIRRELT-2015-42, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

Crainic, Teodor Gabriel, Michael Florian. 2008. National planning models and instruments. *INFOR : Information Systems and Operational Research* **46**(4) 299–308.

Crainic, Teodor Gabriel, Antonio Frangioni, Bernard Gendron. 2001. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics* **112**(1) 73 – 99.

Crainic, Teodor Gabriel, Xiaorui Fu, Michel Gendreau, Walter Rei, Stein W Wallace. 2011. Progressive hedging-based metaheuristics for stochastic network design. *Networks* **58**(2) 114–124.

Crainic, Teodor Gabriel, Michel Gendreau. 2002. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics* **8**(6) 601–627.

Crainic, Teodor Gabriel, Michel Gendreau, Judith M Farvolden. 2000. A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing* **12**(3) 223–236.

Crainic, Teodor Gabriel, Mike Hewitt, Walter Rei. 2014a. Partial decomposition strategies for two-stage stochastic integer programs. Publication CIRRELT-2014-13, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

Crainic, Teodor Gabriel, Mike Hewitt, Walter Rei. 2014b. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers & Operations Research* **43** 90–99.

Crainic, Teodor Gabriel, Mike Hewitt, Walter Rei. 2016. Partial Benders decomposition strategies for two-stage stochastic integer programs. Publication, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

Crainic, Teodor Gabriel, Kap Hwan Kim, et al. 2006a. Intermodal transportation. *Transportation* **14** 467–537.

Crainic, Teodor Gabriel, Gilbert Laporte. 1997. Planning models for freight transportation. *European Journal of Operational Research* **97**(3) 409 – 438.

Crainic, Teodor Gabriel, Bertrand Le Cun, Catherine Roucairol. 2006b. Parallel Branch-and-Bound algorithms. El-Ghazali Talbi, ed., *Parallel Combinatorial Optimization*, chap. 1. John Wiley & Sons, 1–28.

Crainic, Teodor Gabriel, Michel Toulouse. 1998. Parallel metaheuristics. Teodor Gabriel Crainic, Gilbert Laporte, eds., *Fleet Management and Logistics*. Springer US, Boston, MA, 205–251.

Crainic, T.G. 2017. Parallel Meta-Heuristic Search. Marti, R., Psrdalos, P.M., Resende, M.G.C., eds., *Handbook of Heuristics*. Springer, New York.

Dantzig, G., R. Fulkerson, S. Johnson. 1954. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America* **2**(4) 393–410.

Dantzig, George. 1998. *Linear programming and extensions*. Princeton university press.

Dantzig, George B, James K Ho, Gerd Infanger. 1991. Solving stochastic linear programs on a hypercube multicomputer. Tech. Rep. ADA240443, DTIC Document.

de Camargo, Ricardo Saraiva, Gilberto de Miranda Jr., Ricardo P.M. Ferreira. 2011. A hybrid outer-approximation/Benders decomposition algorithm for the single allocation hub location problem under congestion. *Operations Research Letters* **39**(5) 329 – 337.

de Sá, Elisangela Martins, Ricardo Saraiva de Camargo, Gilberto de Miranda. 2013. An improved Benders decomposition algorithm for the tree of hubs location problem. *European Journal of Operational Research* **226**(2) 185 – 202.

Dempster, Michael AH, RT Thompson. 1998. Parallelization and aggregation of nested Benders decomposition. *Annals of Operations Research* **81** 163–188.

Deng, Yan, Shabbir Ahmed, Siqian Shen. 2018. Parallel scenario decomposition of risk-averse 0-1 stochastic programs. *INFORMS Journal on Computing* **30**(1) 90–105.

Dupačová, Jitka, Giorgio Consigli, Stein W. Wallace. 2000. Scenarios for multistage stochastic programs. *Annals of Operations Research* **100**(1) 25–53.

Easwaran, Gopalakrishnan, Halit Üster. 2009. Tabu search and Benders decomposition approaches for a capacitated closed-loop supply chain network design problem. *Transportation Science* **43**(3) 301–320.

Emami, Saeed, Ghasem Moslehi, Mohammad Sabbagh. 2016. A Benders decomposition approach for order acceptance and scheduling problem : a robust optimization approach. *Computational and Applied Mathematics* 1–45.

Eremin, Andrew, Mark Wallace. 2001. Hybrid Benders decomposition algorithms in constraint logic programming. *In Proceding of the 7th International Conference on Principles and Practice of Constraint Programming*. CP '01, Springer-Verlag, Paphos, Cyprus, 1–15.

Errico, Fausto, Teodor Gabriel Crainic, Federico Malucelli, Maddalena Nonato. 2016. A Benders decomposition approach for the symmetric TSP with generalized latency arising in the design of semiflexible transit systems. *Transportation Science* 1–17.

Fábián, Csaba I. 2000. Bundle-type methods for inexact data. *Central European Journal of Operations Research* **8**(1) 35–55.

Fábián, Csaba I., Zoltán Szőke. 2007. Solving two-stage stochastic programming problems with level decomposition. *Computational Management Science* **4**(4) 313–353.

Fischetti, Matteo, Ivana Ljubic, Markus Sinnl. 2016. Redesigning Benders decomposition for large-scale facility location. *Management Science* doi :10.1287/mnsc.2016.2461.

Fischetti, Matteo, Domenico Salvagnin, Arrigo Zanette. 2010. A note on the selection of Benders' cuts. *Mathematical Programming* **124**(1-2) 175–182.

Fisher, Marshall L. 2004. The lagrangian relaxation method for solving integer programming problems. *Manage. Sci.* **50**(12 Supplement) 1861–1871.

Flippo, Olaf E., Alexander H. G. Rinnooy Kan. 1993. Decomposition in general mathematical programming. *Mathematical Programming* **60**(1) 361–382.

Fontaine, Pirmin, Stefan Minner. 2014. Benders decomposition for discrete–continuous linear bilevel problems with application to traffic network design. *Transportation Research Part B : Methodological* **70** 163–172.

Fortz, B., M. Poss. 2009. An improved Benders decomposition applied to a multi-layer network design problem. *Operations Research Letters* **37**(5) 359–364.

Fragkogios, Antonios, Georgios KD Saharidis. 2018. Latest advances on Benders decomposition. *Encyclopedia of Information Science and Technology, Fourth Edition*. IGI Global, 5411–5421.

Frangioni, Antonio, Bernard Gendron. 2009. 0–1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics* **157**(6) 1229 – 1241.

Fukuda, Komei, Thomas M Liebling, Francois Margot. 1997. Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. *Computational Geometry* **8**(1) 1–12.

Gabrel, V., A. Knippel, M. Minoux. 1999. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters* **25**(1) 15–23.

Gelareh, Shahin, Rahimeh Neamatian Monemi, Stefan Nickel. 2015. Multi-period hub location problems in transportation. *Transportation Research Part E : Logistics and Transportation Review* **75** 67–94.

Gendron, Bernard. 2011. Decomposition methods for network design. *Procedia - Social and Behavioral Sciences* **20** 31 – 37.

Gendron, Bernard, Teodor Gabriel Crainic. 1994. Parallel branch-and-branch algorithms : Survey and synthesis. *Operations Research* **42**(6) 1042–1066.

Gendron, Bernard, Teodor Gabriel Crainic, Antonio Frangioni. 1999. Telecommunications network planning. chap. Multicommodity Capacitated Network Design. Springer US, Boston, MA, 1–19.

Gendron, Bernard, Luis Gouveia. 2017. Reformulations by discretization for piecewise linear integer multicommodity network flow problems. *Transportation Science* **51**(2) 629–649.

Gendron, Bernard, Saïd Hanafi, Raca Todosijević. 2018. Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design. *European Journal of Operational Research* –.

Gendron, Bernard, Maria Grazia Scutellà, Rosario G. Garroppo, Gianfranco Nencioni, Luca Tavanti. 2016. A branch-and-benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research* **255**(1) 151 – 162.

Geoffrion, Arthur M. 1970a. Elements of large-scale mathematical programming : Part I : Concepts. *Management Science* **16**(11) 652–675.

Geoffrion, Arthur M. 1970b. Elements of large scale mathematical programming : Part II : Synthesis of algorithms and bibliography. *Management Science* **16**(11) 676–691.

Geoffrion, Arthur M. 1972. Generalized Benders decomposition. *Journal of Optimization Theory and Applications* **10**(4) 237–260.

Geoffrion, Arthur M, Glenn W Graves. 1974. Multicommodity distribution system design by Benders decomposition. *Management Science* **20**(5) 822–844.

Ghamlouche, Ilfat, Teodor Gabriel Crainic, Michel Gendreau. 2003. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations Research* **51**(4) 655–667.

Goffin, J. L., A. Haurie, J. P. Vial. 1992. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science* **38**(2) 284–302.

Gomory, Ralph E. 1958. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society* **64**(3) 275–278.

Grossmann, Ignacio E. 2002. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering* **3**(3) 227–252.

Grothey, A, S Leyffer, KIM McKinnon. 1999. A note on feasibility in Benders decomposition. Numerical Analysis Report NA/188, University of Dundee, Nethergate, Dundee, Scotland, UK.

Günlük, Oktay. 1999. A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming* **86**(1) 17–39.

Harjunkoski, Iiro, Ignacio E Grossmann. 2001. A decomposition approach for the scheduling of a steel plant production. *Computers & Chemical Engineering* **25**(11) 1647–1660.

Harjunkoski, Iiro, Ignacio E Grossmann. 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers & Chemical Engineering* **26**(11) 1533–1552.

Held, Michael, Philip Wolfe, Harlan P. Crowder. 1974. Validation of subgradient optimization. *Mathematical Programming* **6**(1) 62–88.

Hewitt, Mike, George L. Nemhauser, Martin W. P. Savelsbergh. 2010. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing* **22**(2) 314–325.

Higle, Julia L., Suvrajeet Sen. 1991. Stochastic decomposition : An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research* **16**(3) 650–669.

Holloway, Charles A. 1973. A generalized approach to dantzig-wolfe decomposition for concave programs. *Operations Research* **21**(1) 210–220.

Holmberg, Kaj. 1990. On the convergence of cross decomposition. *Mathematical Programming* **47**(1-3) 269–296.

Holmberg, Kaj. 1994. On using approximations of the Benders master problem. *European Journal of Operational Research* **77**(1) 111–125.

Hooker, J. N. 2005. A hybrid method for the planning and scheduling. *Constraints* **10**(4) 385–401.

Hooker, John. 2011. *Logic-based methods for optimization : combining optimization and constraint satisfaction*. John Wiley & Sons.

Hooker, John N. 2007. Planning and scheduling by logic-based Benders decomposition. *Operations Research* **55**(3) 588–602.

Hooker, John N, Greger Ottosson. 2003. Logic-based Benders decomposition. *Mathematical Programming* **96**(1) 33–60.

Jain, Vipul, Ignacio E Grossmann. 2001. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on computing* **13**(4) 258–276.

Jeihoonian, Mohammad, Masoumeh Kazemi Zanjani, Michel Gendreau. 2016. Accelerating Benders decomposition for closed-loop supply chain network design : Case of used durable products with different quality levels. *European Journal of Operational Research* **251**(3) 830–845.

Jenabi, M, SMT Fatemi Ghomi, SA Torabi, SH Hosseinian. 2015. Acceleration strategies of Benders decomposition for the security constraints power system expansion planning. *Annals of Operations Research* **235**(1) 337–369.

Jiang, W., L. Tang, S. Xue. 2009. A hybrid algorithm of tabu search and Benders decomposition for multi-product production distribution network design. *Proceedings of the IEEE International Conference on Automation and Logistics*. ICAL '09, Shenyang, China, 79–84.

Jojic, Vladimir, Stephen Gould, Daphne Koller, et al. 2010. Accelerated dual decomposition for map inference. *ICML*. 503–510.

Jünger, Michael, Thomas M Liebling, Denis Naddef, George L Nemhauser, William R Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, Laurence A Wolsey. 2009. *50 years of integer programming 1958-2008 : From the early years to the state-of-the-art*. Springer Science & Business Media.

Kall, P. 1994. Solution methods in stochastic programming. Jacques Henry, Jean-Pierre Yvon, eds., *System Modelling and Optimization : Proceedings of the 16th IFIP-TC7 Conference, Compiègne, France — July 5–9, 1993*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–22.

Kall, Peter, Stein W Wallace, Peter Kall. 1994. *Stochastic programming*. Springer.

Kaut, Michal, Stein W. Wallace. 2003. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization* **3** 1–16.

Kewcharoenwong, Panitan, Halit Üster. 2014. Benders decomposition algorithms for the fixed-charge relay network design in telecommunications. *Telecommunication Systems* **56**(4) 441–453.

Klibi, Walid, Alain Martel, Adel Guitouni. 2010. The design of robust value-creating supply chain networks : A critical review. *European Journal of Operational Research* **203**(2) 283 – 293.

Kliewer, Georg, Larissa Timajev. 2005. Relax-and-cut for capacitated network design. Gerth Stølting Brodal, Stefano Leonardi, eds., *Algorithms – ESA 2005 : 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005. Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, 47–58.

Koko, Jonas. 2013. A survey on dual decomposition methods. *SeMA Journal* **62**(1) 27–59.

Kouvelis, Panos, Gang Yu. 1997. *Robust Discrete Optimization and Its Applications*, chap. Robust Uncapacitated Network Design and International Sourcing Problems. Springer US, Boston, MA, 290–332.

Kudela, J., P. Popela. 2015. Two-stage stochastic facility location problem : GA with Benders decomposition. *Mendel* 53–58.

Küçükyavuz, Simge, Suvrajeet Sen. 2017. *An Introduction to Two-Stage Stochastic Mixed-Integer Programming*, chap. 1. INFORMS, 1–27.

Lahrichi, Nadia, Teodor Gabriel Crainic, Michel Gendreau, Walter Rei, Gloria Cerasela Crişan, Thibaut Vidal. 2015. An integrative cooperative search framework for multi-decision-attribute combinatorial optimization : Application to the MDPVRP. *European Journal of Operational Research* **246**(2) 400 – 412.

Lai, Ming-Che, Han-Suk Sohn, Tzu-Liang Tseng, Dennis L Bricker. 2012. A hybrid Benders/genetic algorithm for vehicle routing and scheduling problem. *International Journal of Industrial Engineering* **19**(1) 33–46.

Lai, Ming-Che, Han-suk Sohn, Tzu-Liang Bill Tseng, Chunkuan Chiang. 2010. A hybrid algorithm for capacitated plant location problem. *Expert Systems with Applications* **37**(12) 8599–8605.

Land, A. H., A. G. Doig. 1960. An automatic method of solving discrete programming problems. *Econometrica* **28**(3) 497–520.

Laporte, Gilbert. 1992. The vehicle routing problem : An overview of exact and approximate algorithms. *European Journal of Operational Research* **59**(3) 345 – 358.

Laporte, Gilbert, François V Louveaux. 1993. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* **13**(3) 133–142.

Laporte, Gilbert, Francois V Louveaux, Hélene Mercure. 1994. A priori optimization of the probabilistic traveling salesman problem. *Operations Research* **42**(3) 543–549.

Latorre, Jesús M, Santiago Cerisola, Andrés Ramos, Rafael Palacios. 2009. Analysis of stochastic problem decomposition algorithms in computational grids. *Annals of Operations Research* **166**(1) 355–373.

Lemaréchal, Claude, Arkadii Nemirovskii, Yurii Nesterov. 1995. New variants of bundle methods. *Mathematical Programming* **69**(1-3) 111–147.

Li, Xiang. 2013. Parallel nonconvex generalized Benders decomposition for natural gas production network planning under uncertainty. *Computers & Chemical Engineering* **55** 97–108.

Lim, Churlzu. 2010. Relationship among Benders, Dantzig-Wolfe, and Lagrangian optimization. James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, J. Cole Smith, eds., *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons.

Lin, H., H. Üster. 2014. Exact and heuristic algorithms for data-gathering cluster-based wireless sensor network design problem. *IEEE/ACM Transactions on Networking* **22**(3) 903–916.

Lin, Sifeng, Gino J. Lim, Jonathan F. Bard. 2016. Benders decomposition and an IP-based heuristic for selecting IMRT treatment beam angles. *European Journal of Operational Research* **251**(3) 715 – 726.

Linderoth, J. T., M. W. P. Savelsbergh. 1999. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing* **11**(2) 173–187.

Linderoth, Jeff, Stephen Wright. 2003. Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications* **24**(2-3) 207–250.

Linderoth, Jeffrey T. 1998. Topics in parallel integer optimization. Ph.D. thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Little, John D. C., Katta G. Murty, Dura W. Sweeney, Caroline Karel. 1963. An algorithm for the traveling salesman problem. *Operations Research* **11**(6) 972–989.

Lorenz, Ulf, Jan Wolf. 2015. Solving multistage quantified linear optimization problems with the alpha–beta nested Benders decomposition. *EURO Journal on Computational Optimization* **3**(4) 349–370. doi :10.1007/s13675-015-0038-7.

Louveaux, F. V. 1986. Discrete stochastic location models. *Annals of Operations Research* **6**(2) 21–34.

Louveaux, François V., Rüdiger Schultz. 2003. Stochastic integer programming. *Stochastic Programming*, *Handbooks in Operations Research and Management Science*, vol. 10. Elsevier, 213 – 266.

Lulli, Guglielmo, Suvrajeet Sen. 2004. A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems. *Management Science* **50**(6) 786–796.

Luong, Curtiss. 2015. An examination of Benders decomposition approaches in large-scale healthcare optimization problems. Master's thesis, University of Toronto.

Maculan, Nelson, Marcos M. Passini, José André M. Brito, Abdel Lisser. 2002. Column generation method for network design. Michel Gendreau, Patrice Marcotte, eds., *Transportation and Network Analysis : Current Trends : Miscellanea in honor of Michael Florian*. Springer US, Boston, MA, 165–179.

Maggioni, Francesca, Elisabetta Allevi, Marida Bertocchi. 2016. Monotonic bounds in multistage mixed-integer stochastic programming. *Computational Management Science* **13**(3) 423–457.

Magnanti, T. L., P. Mireault, R. T. Wong. 1986. Tailoring Benders decomposition for uncapacitated network design. *Mathematical programming study* **26** 112–154.

Magnanti, Thomas L, Robert W Simpson. 1978. Transportation network analysis and decomposition methods. Report no. DOT-TSC-RSPD-78-6, U.S. Department of Transportation.

Magnanti, Thomas L, Richard T Wong. 1981. Accelerating Benders decomposition : Algorithmic enhancement and model selection criteria. *Operations Research* **29**(3) 464–484.

Magnanti, Thomas L, Richard T Wong. 1984. Network design and transportation planning : Models and algorithms. *Transportation Science* **18**(1) 1–55.

Maravelias, Christos T., Ignacio E. Grossmann. 2004. Using MILP and CP for the scheduling of batch chemical processes. Jean-Charles Régin, Michel Rueher, eds., *First International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. CPAIOR '04, Springer, Nice, France, 1–20.

McDaniel, Dale, Mike Devine. 1977. A modified Benders' partitioning algorithm for mixed integer programming. *Management Science* **24**(3) 312–319.

Mercier, Anne, Jean-François Cordeau, François Soumis. 2005. A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research* **32**(6) 1451–1476.

Mercier, Anne, François Soumis. 2007. An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research* **34**(8) 2251 – 2265.

Meyer, R. R. 1974. On the existence of optimal solutions to integer and mixed-integer programming problems. *Mathematical Programming* **7**(1) 223–235.

Minoux, M. 1986. *Mathematical programming : theory and algorithms*. Wiley-Interscience series in discrete mathematics and optimization, Wiley.

Minoux, Michel. 1984. Subgradient optimization and Benders decomposition for large scale programming. R. W. Cottle, M. L. Kelmanson, B. Korte, eds., *Mathematical Programming*. North Holland Amsterdam, 271–288.

Minoux, Michel. 2001. Discrete cost multicommodity network optimization problems and exact solution methods. *Annals of Operations Research* **106**(1) 19–46.

MirHassani, S.A., C. Lucas, G. Mitra, E. Messina, C.A. Poojari. 2000. Computational solution of capacity planning models under uncertainty. *Parallel Computing* **26**(5) 511 – 538.

Mitra, Sumit, Pablo Garcia-Herreros, Ignacio E. Grossmann. 2016. A cross-decomposition scheme with integrated primal–dual multi-cuts for two-stage stochastic programming investment planning problems. *Mathematical Programming* **157**(1) 95–119. doi :10.1007/s10107-016-1001-y.

Moreno-Centeno, Erick, Richard M Karp. 2013. The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment. *Operations Research* **61**(2) 453–468.

Moritsch, Hans W., G. Ch. Pflug, M. Siomak. 2001. Asynchronous nested optimization algorithms and their parallel implementation. *Wuhan University Journal of Natural Sciences* **6**(1) 560–567.

Muter, İbrahim, Ş İlker Birbil, Kerem Bülbül. 2015. Benders decomposition and column-and-row generation for solving large-scale linear programs with column-dependent-rows. *Optimization Online, Available at http ://www.optimization-online.org/DB_FILE/2015/11/5184.pdf* .

Naoum-Sawaya, Joe, Samir Elhedhli. 2010. A nested Benders decomposition approach for telecommunication network planning. *Naval Research Logistics (NRL)* **57**(6) 519–539. doi : 10.1002/nav.20419.

Naoum-Sawaya, Joe, Samir Elhedhli. 2013. An interior-point Benders based branch-and-cut algorithm for mixed integer programs. *Annals of Operations Research* **210**(1) 33–55.

Nemhauser, G. L., W. B. Widhelm. 1971. A modified linear program for columnar methods in mathematical programming. *Operations Research* **19**(4) 1051–1060.

Nemhauser, George L., Laurence A. Wolsey. 1988. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA.

Nemirovski, A., A. Juditsky, G. Lan, A. Shapiro. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization* **19**(4) 1574–1609.

Newman, Alexandra M., Martin Weiss. 2013. A survey of linear and mixed-integer optimization tutorials. *INFORMS Transactions on Education* **14**(1) 26–38.

Nielsen, Soren S, Stavros A Zenios. 1997. Scalable parallel Benders decomposition for stochastic linear programming. *Parallel Computing* **23**(8) 1069–1088.

Ntaimo, Lewis. 2010. Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations Research* **58**(1) 229–243.

O'Kelly, Morton E., Henrique Pacca L. Luna, Ricardo S. Camargo, Gilberto Miranda. 2014. Hub location problems with price sensitive demands. *Networks and Spatial Economics* **15**(4) 917–945.

Oliveira, Fabricio, Ignacio E Grossmann, Silvio Hamacher. 2014. Accelerating Benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain. *Computers & Operations Research* **49** 47–58.

Osman, Hany, MF Baki. 2014. Balancing transfer lines using Benders decomposition and ant colony optimisation techniques. *International Journal of Production Research* **52**(5) 1334–1350.

Pacqueau, Remi, SoumisS Francois, Hoang Le Nguyen. 2012. A fast and accurate algorithm for stochastic integer programming, appllied to stochastic shift scheduling. Publication G-2012-29, Groupe d'études et de recherche en analyse des décisions (GERAD), Université de Montréal, Montréal, QC, Canada.

Pan, Feng, David P. Morton. 2008. Minimizing a stochastic maximum-reliability path. *Networks* **52**(3) 111–119.

Papadakos, Nikolaos. 2008. Practical enhancements to the magnanti–wong method. *Operations Research Letters* **36**(4) 444–449.

Papadakos, Nikolaos. 2009. Integrated airline scheduling. *Computers & Operations Research* **36**(1) 176–195.

Pérez-Galarce, Francisco, Eduardo Álvarez-Miranda, Alfredo Candia-Véjar, Paolo Toth. 2014. On exact solutions for the minmax regret spanning tree problem. *Computers & Operations Research* **47** 114–122.

Peterson, Benjamin, Michael A. Trick. 2009. A Benders' approach to a transportation network design problem. *Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. CPAIOR '09, Springer-Verlag, Berlin, Heidelberg, 326–327.

Pfeiffer, Laurent, Romain Apparigliato, Sophie Auchapt. 2012. Two methods of pruning Benders' cuts and their application to the management of a gas portfolio. *Optimization Online, Available at http ://www.optimization-online.org/DB_FILE/2012/11/3683.pdf* .

Pinheiro, Placido Rogerio, Paulo Roberto Oliveira. 2013. A hybrid approach of bundle and Benders applied large mixed linear integer problem. *Journal of Applied Mathematics* **2013** 11 pages.

Pishvaee, MS, J Razmi, SA Torabi. 2014. An accelerated Benders decomposition algorithm for sustainable supply chain network design under uncertainty : A case study of medical needle and syringe supply chain. *Transportation Research Part E : Logistics and Transportation Review* **67** 14–38.

Poojari, Chandra A, John E Beasley. 2009. Improving Benders decomposition using a genetic algorithm. *European Journal of Operational Research* **199**(1) 89–97.

Qi, Yutao, Zhanting Hou, He Li, Jianbin Huang, Xiaodong Li. 2015. A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows. *Computers & Operations Research* **62**(Supplement C) 61 – 77.

Rahmaniani, Ragheb, Teodor Gabriel Crainic, Michel Gendreau, Walter Rei. 2017a. The Benders decomposition algorithm : A literature review. *European Journal of Operational Research* **259**(3) 801 – 817.

Rahmaniani, Ragheb, Teodor Gabriel Crainic, Michel Gendreau, Walter Rei. 2017b. A Benders decomposition method for two-stage stochastic network design problems. Publication CIRRELT-2017-22, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

Rahmaniani, Ragheb, Abdolsalam Ghaderi. 2013. A combined facility location and network design problem with multi-type of capacitated links. *Applied Mathematical Modelling* **37**(9) 6400 – 6414.

Rahmaniani, Ragheb, Gona Rahmaniani, Armin Jabbarzadeh. 2014. Variable neighborhood search based evolutionary algorithm and several approximations for balanced location–allocation design problem. *The International Journal of Advanced Manufacturing Technology* **72**(1) 145–159.

Raidl, Günther R. 2015. Decomposition based hybrid metaheuristics. *European Journal of Operational Research* **244**(1) 66–76.

Raidl, Günther R., Thomas Baumhauer, Bin Hu. 2014. Speeding up logic-based Benders' decomposition by a metaheuristic for a bi-level capacitated vehicle routing problem. Maria J. Blesa, Christian Blum, Stefan Voß, eds., *9th International Workshop on Hybrid Metaheuristics*. HM 2014, Springer International Publishing, Hamburg, Germany, 183–197.

Rais, Abdur, Ana Viana. 2011. Operations research in healthcare : a survey. *International Transactions in Operational Research* **18**(1) 1–31.

Ralphs, T.K., L. Ladányi, M.J. Saltzman. 2003. Parallel branch, cut, and price for large-scale discrete optimization. *Mathematical Programming* **98**(1) 253–280.

Randazzo, C. D., H. P. L. Luna, P. Mahey. 2001. Benders decomposition for local access network design with two technologies. *Discrete Mathematics and Theoretical Computer Science* **4**(2) 235–246.

Rei, Walter, Jean-François Cordeau, Michel Gendreau, Patrick Soriano. 2009. Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing* **21**(2) 333–345.

Restrepo, María I, Bernard Gendron, Louis-Martin Rousseau. 2015. Combining Benders decomposition and column generation for multi-activity tour scheduling. Publication CIRRELT-2015-57, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Rockafellar, R. T. 1970. *Convex Analysis*, *Princeton Math. Series*, vol. 28. Princeton University Press. URL `http://www.math.washington.edu/~rtr/papers.html`.

Rockafellar, R Tyrrell, Roger J-B Wets. 1991. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research* **16**(1) 119–147.

Roshanaei, Vahid, Curtiss Luong, Dionne M. Aleman, David Urbach. 2017. Propagating logic-based Benders decomposition approaches for distributed operating room scheduling. *European Journal of Operational Research* **257**(2) 439–455.

Roussel, Sylvie, Jacques A Ferland, Lorena Pradenas. 2004. Improving Benders decomposition to solve the tree-bucking problem. Working paper, Available at http ://www.iro.umontreal.ca/ ferland/Tutorial/Forestry/Tree_Bucking.pdf.

Rubiales, AJ, PA Lotito, LA Parente. 2013. Stabilization of the generalized Benders decomposition applied to short-term hydrothermal coordination problem. *IEEE Latin America Transactions* **11**(5) 1212–1224.

Rubinstein, Reuven Y., Alexander Shapiro. 1990. Optimization of static simulation models by the score function method. *Mathematics and Computers in Simulation* **32**(4) 373 – 392.

Rush, Alexander M., Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *J. Artif. Int. Res.* **45**(1) 305–362.

Ruszczyński, Andrzej. 1986. A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming* **35**(3) 309–333.

Ruszczyński, Andrzej. 1997. Decomposition methods in stochastic programming. *Mathematical Programming* **79**(1) 333–353.

Ruszczyński, Andrzej. 1999. Some advances in decomposition methods for stochastic linear programming. *Annals of Operations Research* **85** 153–172.

Ruszczyński, Andrzej. 2003. Decomposition methods. Andrzej Ruszczyński, A. Shapiro, eds., *Stochastic Programming*, *Handbooks in Operations Research and Management Science*, vol. 10. Elsevier, 141–211.

Ruszczyński, Andrzej, Artur Świętanowski. 1997. Accelerating the regularized decomposition method for two stage stochastic linear problems. *European Journal of Operational Research* **101**(2) 328–342.

Saharidis, Georges K, Marianthi G Ierapetritou. 2009. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization* **44**(1) 29–51.

Saharidis, Georgios KD, Maria Boile, Sotiris Theofanis. 2011. Initialization of the Benders master problem using valid inequalities applied to fixed-charge network problems. *Expert Systems with Applications* **38**(6) 6627–6636.

Saharidis, Georgios KD, Marianthi G Ierapetritou. 2010. Improving Benders decomposition using maximum feasible subsystem (MFS) cut generation strategy. *Computers & Chemical Engineering* **34**(8) 1237–1245.

Saharidis, Georgios KD, Marianthi G Ierapetritou. 2013. Speed-up Benders decomposition using maximum density cut (MDC) generation. *Annals of Operations Research* **210**(1) 101–123.

Saharidis, Georgios KD, Michel Minoux, Marianthi G Ierapetritou. 2010. Accelerating Benders method using covering cut bundle generation. *International Transactions in Operational Research* **17**(2) 221–237.

Sahinidis, NV, Ignacio E Grossmann. 1991. Convergence properties of generalized Benders decomposition. *Computers & Chemical Engineering* **15**(7) 481–491.

Santini, Alberto, Christian E.M. Plum, Stefan Ropke. 2018. A branch-and-price approach to the feeder network design problem. *European Journal of Operational Research* **264**(2) 607 – 622.

Santoso, Tjendera, Shabbir Ahmed, Marc Goetschalckx, Alexander Shapiro. 2005. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research* **167**(1) 96–115.

Schütz, Peter, Asgeir Tomasgard, Shabbir Ahmed. 2009. Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research* **199**(2) 409 – 419.

Sellmann, Meinolf, Georg Kliewe, Achim Kobe stein. 2002. Lagrangian cardinality cuts and variable fixing for capacitated network design. Rolf Möhring, Rajeev Raman, eds., *Algorithms — ESA 2002 : 10th Annual European Symposium Rome, Italy, September 17–21, 2002 Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, 845–858.

Sen, Goutam, Mohan Krishnamoorthy, Narayan Rangaraj, Vishnu Narayanan. 2015. Exact approaches for static data segment allocation problem in an information network. *Computers & Operations Research* **62** 282–295.

Sen, Suvrajeet. 1993. Subgradient decomposition and differentiability of the recourse function of a two stage stochastic linear program. *Operations Research Letters* **13**(3) 143 – 148.

Sen, Suvrajeet, Julia L. Higle. 1999. An introductory tutorial on stochastic linear programming models. *Interfaces* **29**(2) 33–61.

Sen, Suvrajeet, Julia L Higle. 2005. The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming : set convexification. *Mathematical Programming* **104**(1) 1–20.

Sen, Suvrajeet, Hanif D Sherali. 2006. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming* **106**(2) 203–223.

Shapiro, Alexander, Darinka Dentcheva, Andrzej Ruszczyński. 2014. *Lectures on stochastic programming : modeling and theory*, vol. 16. SIAM.

Sherali, Hanif D, Barbara MP Fraticelli. 2002. A modification of Benders' decomposition algorithm for discrete subproblems : An approach for stochastic programs with integer recourse. *Journal of Global Optimization* **22**(1-4) 319–342.

Sherali, Hanif D, Brian J Lunday. 2013. On generating maximal nondominated Benders cuts. *Annals of Operations Research* **210**(1) 57–72.

Simsek, Koray D. 2008. Introduction to stochastic programming and its applications to finance. *Handbook of Finance*. John Wiley & Sons.

Talbi, El-Ghazali. 2006. *Parallel combinatorial optimization*, vol. 58. John Wiley & Sons.

Tang, Lixin, Wei Jiang, Georgios KD Saharidis. 2013. An improved Benders decomposition algorithm for the logistics facility location problem with capacity expansions. *Annals of Operations Research* **210**(1) 165–190.

Tarvin, D. Antony, R. Kevin Wood, Alexandra M. Newman. 2016. Benders decomposition : Solving binary master problems by enumeration. *Operations Research Letters* **44**(1) 80 – 85.

Taşkın, Z Caner, Mucahit Cevik. 2013. Combinatorial Benders cuts for decomposing IMRT fluence maps using rectangular apertures. *Computers & Operations Research* **40**(9) 2178–2186.

Thorsteinsson, Erlendur S. 2001. Branch-and-check : A hybrid framework integrating mixed integer programming and constraint logic programming. *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*. CP '01, Springer-Verlag, London, UK, UK, 16–30.

Trukhanov, Svyatoslav, Lewis Ntaimo, Andrew Schaefer. 2010. Adaptive multicut aggregation for two-stage stochastic linear programs with recourse. *European Journal of Operational Research* **206**(2) 395–406.

Uryasev, Stanislav, Panos M Pardalos. 2013. *Stochastic optimization : algorithms and applications*, vol. 54. Springer Science & Business Media.

van Ackooij, W, A Frangioni, W de Oliveira. 2015. Inexact stabilized Benders' decomposition approaches to chance-constrained problems with finite support. *Applied Mathematics and Computation* **270** 193–215.

Van Roy, Tony J. 1983. Cross decomposition for mixed integer programming. *Mathematical programming* **25**(1) 46–63.

Van Slyke, Richard M, Roger Wets. 1969. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics* **17**(4) 638–663.

Verstichel, Jannes, Joris Kinable, Patrick De Causmaecker, G Vanden Berghe. 2015. A combinatorial Benders decomposition for the lock scheduling problem. *Computers & Operations Research* **54** 117–128.

Vladimirou, Hercules. 1998. Computational assessment of distributed decomposition methods for stochastic linear programs. *European Journal of Operational Research* **108**(3) 653–670.

Wallace, S., W. Ziemba. 2005. *Applications of Stochastic Programming*. Society for Industrial and Applied Mathematics.

Wallace, Stein W., Stein-Erik Fleten. 2003. Stochastic programming models in energy. *Stochastic Programming*, *Handbooks in Operations Research and Management Science*, vol. 10. Elsevier, 637 – 677.

Wang, Qin, James D McCalley, Tongxin Zheng, Eugene Litvinov. 2016a. Solving corrective risk-based security-constrained optimal power flow with Lagrangian relaxation and Benders decomposition. *International Journal of Electrical Power & Energy Systems* **75** 255–264.

Wang, Xin, Teodor Gabriel Crainic, Stein Wallace. 2016b. Stochastic scheduled service network design : The value of deterministic solutions. Publication CIRRELT-2016-14, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal.

Wentges, Paul. 1996. Accelerating Benders' decomposition for the capacitated facility location problem. *Mathematical Methods of Operations Research* **44**(2) 267–290.

Wets, R. 1983. Stochastic programming : Solution techniques and approximation schemes. Achim Bachem, Bernhard Korte, Martin Grötschel, eds., *Mathematical Programming The State of the Art : Bonn 1982*. Springer Berlin Heidelberg, Berlin, Heidelberg, 566–603.

Wets, Roger J-B. 2002. Stochastic programming models : Wait-and-see versus here-and-now. Claude Greengard, Andrzej Ruszczynski, eds., *Decision Making Under Uncertainty : Energy and Power*. Springer New York, New York, NY, 1–15.

Wheatley, David, Fatma Gzara, Elizabeth Jewkes. 2015. Logic-based Benders decomposition for an inventory-location problem with service constraints. *Omega* **55** 10–23.

Wieberneit, Nicole. 2008. Service network design for freight transportation : a review. *OR Spectrum* **30**(1) 77–112.

Wolf, Christian. 2014. Advanced acceleration techniques for nested Benders decomposition in stochastic programming. Ph.D. thesis, Universität Paderborn.

Wolf, Christian, Achim Koberstein. 2013. Dynamic sequencing and cut consolidation for the parallel hybrid-cut nested L-shaped method. *European Journal of Operational Research* **230**(1) 143–156.

Wu, Peiling, Joseph C Hartman, George R Wilson. 2003. A demand-shifting feasibility algorithm for Benders decomposition. *European Journal of Operational Research* **148**(3) 570–583.

Yang, Huasheng, Jatinder ND Gupta, Lina Yu, Li Zheng. 2016. An improved L-shaped method for solving process flexibility design problems. *Mathematical Problems in Engineering* **2016**.

Yang, Yu, Jong Min Lee. 2012. A tighter cut generation strategy for acceleration of Benders decomposition. *Computers & Chemical Engineering* **44** 84–93.

yong Yu, Li, Xiao dong Ji, Shou-Yang Wang. 2003. Stochastic programming models in financial optimization : A survey. *Advanced Modeling and Optimization* **5**.

Zakeri, Golbon, Andrew B Philpott, David M Ryan. 2000. Inexact cuts in Benders decomposition. *SIAM Journal on Optimization* **10**(3) 643–657.

Zanjani, Masoumeh Kazemi, Daoud Ait-Kadi, Mustapha Nourelfath. 2013. A stochastic programming approach for sawmill production planning. *International Journal of Mathematics in Operational Research* **5**(1) 1–18.

Zaourar, Sofia, Jérôme Malick. 2014. Quadratic stabilization of Benders decomposition. Tech. rep. URL `Availableat,https://hal.archives-ouvertes.fr/hal-01181273/document`. Working paper.

Zarandi, Mohammad Mehdi Fazel. 2010. Using decomposition to solve facility location/fleet management problems. Ph.D. thesis, University of Toronto.

Zhang, Joyce Li, K Ponnambalam. 2006. Hydro energy management optimization in a deregulated electricity market. *Optimization and Engineering* **7**(1) 47–61.

Zhu, Yushan, Takahito Kuno. 2003. Global optimization of nonconvex MINLP by a hybrid branch-and-bound and revised general Benders decomposition approach. *Industrial & Engineering Chemistry Research* **42**(3) 528–539.

Zou, Jikai, Shabbir Ahmed, Xu Andy Sun. 2016. Stochastic dual dynamic integer programming. *Available at, http ://www.optimization-online.org/DB_HTML/2016/05/5436.html* .

Zverovich, Victor, Csaba I Fábián, Eldon FD Ellison, Gautam Mitra. 2012. A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition. *Mathematical Programming Computation* **4**(3) 211–238.

APPENDIX A LOWER BOUND LIFTING INEQUALITY

Consider LP relaxation of the deterministic multicommodity capacitated fixed-charged network design (MCFND) problem associated to scenario $\omega \in \Omega$, denoted $MCNF(\omega)$, which has the following form :

$$(MCNF(\omega)) \quad v(d(\omega)) := \min_{x(\omega)\in R_+^{|A||K|}} \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} \left(c_a^k + \frac{f_a}{u_a}\right) x_a^k(\omega)$$
$$\text{s.t.} \qquad \text{(B.2)} \quad \text{and} \quad \sum_{k\in\mathcal{K}} x_a^k(\omega) \leq u_a \quad \forall a \in \mathcal{A},$$

let $\bar{x}(\omega)$ be the optimal solution of this program. We can derive the design values according to the relation $\sum_{k\in\mathcal{K}} x_a^k(\omega) = u_a y_a, \forall a \in \mathcal{A}$ which is alway satisfied for $y \in [0,1]$. At any optimal solution, according to the property of "wait and see" and "here and now" solutions in stochastic programming (Birge and Louveaux (1997)), following relation among the Benders reformulation of the extensive form and $MCNF(\omega)$ for scenario ω holds : $\sum_{a\in\mathcal{A}} f_a y_a + \theta(\omega) \geq \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} (c_a^k + \frac{f_a}{u_a})\bar{x}_a^k(\omega)$. Accordingly to the variable transformation $\bar{y}_a = \frac{\sum_{k\in\mathcal{K}} x_a^k(\omega)}{u_a}, \forall a \in \mathcal{A}$, we have $\sum_{a\in\mathcal{A}} f_a y_a + \theta(\omega) \geq \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a\in\mathcal{A}} f_a \frac{\sum_{k\in\mathcal{K}} \bar{x}_a^k(\omega)}{u_a} = \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a\in\mathcal{A}} f_a \bar{y}_a$ which gives $\theta(\omega) \geq \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a\in\mathcal{A}} f_a \bar{y}_a - \sum_{a\in\mathcal{A}} f_a y_a = \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a\in\mathcal{A}} f_a (\bar{y}_a - y_a)$.

Solving a MCNF problem for each scenario may not computationally be interesting. Thus, we propose to solve only a single MCNF with minimum demand, i.e., $d_k^{min} = min_{\omega\in\Omega} \{d_k^\omega\}, \forall k \in \mathcal{K}$. The benefit of this auxiliary problem is that its solution gives a valid lower bound for all the SPs. The obtained bound can be further improved for each SP according to 4.1. To do so, we use the sensitivity analysis theory. Let $\bar{\pi}$ be the dual variables associated to the flow conservation constraints and Δ indicate the set of alternative optimal dual solutions. The function $v(d(\omega))$ is piece-wise linear in $d(\omega)$. Thus, $v(d^{min} + \tilde{d}) \geq v(d^{min}) + \max_{\pi\in\Delta} \pi^T \tilde{d} \geq v(d^{min}) + \bar{\pi}^T \tilde{d}$. If we set $\tilde{d} = d(\omega) - d^{min}$ for each SP, we get inequality (4.14).

APPENDIX B FEASIBILITY RESTORATION STRATEGY

The convex combination $y^{ws} = \lambda\bar{y}+(1-\lambda)y^{ws}$ may yield an infeasible y^{ws} solution. To restore y^{ws} to a feasible solution, we solve following linear program :

$$\begin{array}{lll} \min\limits_{\hat{y}\in R_+^{|A|}} & \sum_{a\in\mathcal{A}} f_a\hat{y}_a & \\ \text{s.t.} & \text{(B.2)} \quad \text{and} \quad \sum_{k\in\mathcal{K}} x_a^k(\omega) \leq u_a\left(\bar{y}_a + \hat{y}_a\right) & \forall a \in \mathcal{A} \\ & \hat{y}_a \leq 1 - \bar{y}_a & \forall a \in \mathcal{A}, \end{array}$$

where ω is the SP for which y^{ws} has been infeasible. After solving this problem, y^{ws} is set equal to $\bar{y}_a + \hat{y}_a$.

APPENDIX C LIST OF ACRONYMS

Table C.1 List of acronyms

Acronym	Definition	Acronym	Definition
BD	Benders Decomposition	NCI	Network Connectivity Inequality
CI	Cover Inequality	ND	Network Design
DSP	Dual Sub-Problem	OD	Origin-Destination
FP	Flow Pack	PDS	Partial Decomposition Strategy
LB	Lower Bound	RMP	Restricted Master Problem
LBF	Lower Bounding Function	SDSP	Strong Dual Subproblem
LP	Linear Programming	SI	Strong Inequalities
MCFNDSD	Multi-Commodity Capacitated Fixed-charge Network Design Problem with Stochastic Demands	SND	Stochastic Network Design
MCI	Minimum Cardinality Inequality	SP	Sub-Problem
MCNF	Multi-Commodity Capacitated Network Flow	UB	Upper Bound
MILP	Mixed-Integer Linear Program	VI	Valid Inequality
MP	Master Problem	WS	Warm Start

APPENDIX D PROOF OF PROPOSITION 5.1

To show the validity of the cut we need to demonstrate that the above inequality holds true for any $y \in F \bigcap \mathbb{Z}^n_+$, i.e., it does not cut off any feasible integer solution. We first assume that y^* is an integer feasible solution, i.e., $y^* \in F \bigcap \mathbb{Z}^n_+$. Accordingly, $\theta \geq \min_{(x,z)\in X}\{c^\top x : z = y^*\} = \min_{(x,z)\in X}\{c^\top x : z = y^*,\ z \in \mathbb{Z}^n_+\} \geq \max_{\lambda\in\mathbb{R}^n}\left\{\lambda^\top y^* + \min_{(x,z)\in X}\{c^\top x - \lambda^\top z : z \in \mathbb{Z}^n_+\}\right\}$ for any $y^* \in F \bigcap \mathbb{Z}^n_+ \subseteq F$. We thus deduce that $\theta \geq \lambda^\top y + Q(\lambda)$ for any $\lambda \in \mathbb{R}^n$ is a valid optimality cut at any feasible integer solution, where $Q(\lambda) = \min_{(x,z)\in X}\{c^\top x - \lambda^\top z : z \in \mathbb{Z}^n_+\}$. Thus, if y^* is fractional and $y^* \in F$, the validity of (5.10) follows from setting λ to an arbitrary maximizer of $\max_{\lambda\in\mathbb{R}^n}\left\{\lambda^\top y^* + Q(\lambda)\right\}$.

APPENDIX E PROOF OF THEOREM 5.1

For a given $y^* \in F$ and $\hat{\lambda}$, we have $\theta \geq \max_{\lambda \in \mathbb{R}^n} \left\{ \lambda^\top y^* + \min_{(x,z) \in X : z \in \mathbb{Z}^n_+} \{c^\top x - \lambda^\top z\} \right\} \geq \hat{\lambda}^\top y^* + \min_{(x,z) \in X : z \in \mathbb{Z}^n_+} \{c^\top x - \hat{\lambda}^\top z\} \geq \hat{\lambda}^\top y^* + \min_{(x,z) \in X} \{c^\top x - \hat{\lambda}^\top z\}$. This gives following two optimality cuts : (i) $\theta \geq \hat{\lambda}^\top y + \min_{(x,z) \in X : z \in \mathbb{Z}^n_+} \{c^\top x - \hat{\lambda}^\top z\}$ and (ii) $\theta \geq \hat{\lambda}^\top y + \min_{(x,z) \in X} \{c^\top x - \hat{\lambda}^\top z\}$. We observe that the optimality cut (i) is equal to fixing λ to $\hat{\lambda}$ in (5.9) to generate cut (5.10). While the optimality cut (ii) is equal to the cut that we extract from (5.8) by setting $\lambda = \hat{\lambda}$. The cuts (i) and (ii) are in parallel since they have the same slope λ. However, the constant part in the former is larger by an amount equal to the LP gap of the minimization problem which is equal to σ in (5.11).

APPENDIX F PROOF OF PROPOSITION 5.2

We first recall that following two formulations are equal

$$\min_{(x,z,v)\in\mathbb{R}^m_+\times\mathbb{R}^n_+\times\mathbb{R}^l_+}\{\mathbb{1}^\top v:\ Bz\geq b,\ Wx+Tz+v\geq h,\ z=y^*\}= \tag{F.1}$$

$$\min_{(x,z,v,u)\in\mathbb{R}^m_+\times\mathbb{R}^n_+\times\mathbb{R}^l_+\times\mathbb{R}^k_+}\{\mathbb{1}^\top v+\mathbb{1}^\top u:\ Bz+u\geq b,\ Wx+Tz+v\geq h,\ z=y^*\} \tag{F.2}$$

given that $u=0$ since $Bz\geq b$ is satisfied through $z=y^*$ and y^* is the master solution. Thus, following the same steps as in Proposition (5.1), we directly conclude that (5.13) provides a valid cut for any $\beta\in\mathbb{R}^n$. To show that the cut is violated, notice that $\mathbb{1}^\top\bar{v}>0$ in (F.1) given that y^* is an infeasible solution for which there is no $x\in\mathbb{R}^n_+$ such that $Wx\geq h-Ty^*$. Thus,

$$\begin{aligned}
0<\min_{(x,z,v,u)\in\mathbb{R}^m_+\times\mathbb{R}^n_+\times\mathbb{R}^l_+\times\mathbb{R}^k_+}&\{\mathbb{1}^\top v+\mathbb{1}^\top u:\ Bz+u\geq b,\ Wx+Tz+v\geq h,\ z=y^*\}=\\
\max_{\beta\in\mathbb{R}^n}\ \min_{(x,z,v,u)\in\mathbb{R}^m_+\times\mathbb{R}^n_+\times\mathbb{R}^l_+\times\mathbb{R}^k_+}&\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top(z-y^*):\\
&\qquad Bz+u\geq b,\ Wx+Tz+v\geq h,\ z=y^*\}\leq\\
\max_{\beta\in\mathbb{R}^n}\ \min_{(x,z,v,u)\in\mathbb{R}^m_+\times\mathbb{Z}^n_+\times\mathbb{R}^l_+\times\mathbb{R}^k_+}&\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top(z-y^*):\\
&\qquad Bz+u\geq b,\ Wx+Tz+v\geq h,\ z=y^*\}
\end{aligned}$$

APPENDIX G PROOF OF THEOREM 5.2

First, recall that due to the Theorem 6.2 in Nemhauser and Wolsey (1988)

$$\max_{\lambda\in\mathbb{R}^n}\Big\{\lambda^\top y^* + \min_{(x,z)\in X}\{c^\top x - \lambda^\top z :\ z\in\mathbb{Z}^n_+\}\Big\} \equiv \max_{\lambda\in\mathbb{R}^n}\Big\{\lambda^\top y^* + \min_{(x,z)\in conv(\tilde{X})}\{c^\top x - \lambda^\top z\}\Big\}$$

and

$$\max_{\beta\in\mathbb{R}^n}\Big\{\beta^\top y^* + \min_{(x,z',v,u)\in H}\{\mathbb{1}^\top v + \mathbb{1}^\top u - \beta^\top z' : z'\in\mathbb{Z}^n_+\ \ : z'\in\mathbb{Z}^n_+\}\Big\} \equiv \max_{\beta\in\mathbb{R}^n}\Big\{\beta^\top y^* + \min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v + \mathbb{1}^\top u - \beta^\top z'\}\Big\}$$

where $\tilde{X} = X\cap\{z\in\mathbb{Z}^n_+\}$, $\tilde{H} = H\cap\{z'\in\mathbb{Z}^n_+\}$ and $conv(\Theta)$ represents the convex hull of a given set Θ. Notice that we use z' to represent the copied variables in the feasibility problem in order to distinguish them from the copied variables that are included in the optimality problem. We thus have the following reformulation of the LP relaxation of the MP,

$$\begin{aligned}\min_{\theta,y\in Y}\Big\{f^\top y+\theta :\ \theta \ge \max_{\lambda\in\mathbb{R}^n}\Big\{\lambda^\top y + \min_{(x,z)\in conv(\tilde{X})}\{c^\top x-\lambda^\top z\}\Big\},\\ 0\ge \max_{\beta\in\mathbb{R}^n}\Big\{\beta^\top y + \min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\Big\}\Big\} =\\ \min_{\theta,y\in Y}\Big\{f^\top y+\theta :\ \theta\ge \lambda^\top y + \min_{(x,z)\in conv(\tilde{X})}\{c^\top x-\lambda^\top z\}\ \forall\lambda\in E_o,\\ 0\ge \beta^\top y + \min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\Big\}\end{aligned}$$

where E_o and E_r are the sets of extreme points of the Lagrangian dual programs associated to the optimality and feasibility cuts, respectively. Accordingly, we have the following master

formulation

$$\min_{\theta,y\in Y}\Big\{f^\top y+\theta:\ \theta\geq\lambda^\top y+\min_{(x,z)\in conv(\tilde{X})}\{c^\top x-\lambda^\top z\}\ \forall\lambda\in E_o,$$
$$0\geq\beta^\top y+\min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\Big\}=$$
$$\min_{y\in Y}\Big\{f^\top y+\max_{\lambda\in E_o}\Big\{\lambda^\top y+\min_{(x,z)\in conv(\tilde{X})}\{c^\top x-\lambda^\top z\}\Big\}:$$
$$0\geq\beta^\top y+\min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\Big\}=$$
$$\min_{y\in Y}\Big\{\max_{\lambda\in E_o}\min_{(x,z)\in conv(\tilde{X})}\{f^\top y+c^\top x+\lambda^\top(y-z)\}:$$
$$0\geq\beta^\top y+\min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\Big\}$$

It should be noted that for any function $g:U\times V\to\mathbb{R}$, the following inequality between max-min and min-max operators is always satisfied $\sup_{u\in U}\inf_{v\in V}g(u;v)\leq\inf_{v\in V}\sup_{u\in U}g(z;v)$. Thus,

$$\min_{y\in Y}\Big\{\max_{\lambda\in E_o}\min_{(x,z)\in conv(\tilde{X})}\{f^\top y+c^\top x+\lambda^\top(y-z)\}:$$
$$0\geq\beta^\top y+\min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\Big\}\qquad\geq$$
$$\max_{\lambda\in E_o}\ \min_{y\in Y}\Big\{\min_{(x,z)\in conv(\tilde{X})}\{f^\top y+c^\top x+\lambda^\top(y-z)\}:$$
$$0\geq\beta^\top y+\min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\Big\}\qquad=$$
$$\max_{\lambda\in E_o}\ \min_{y\in Y,(x,z)\in conv(\tilde{X})}\Big\{f^\top y+c^\top x+\lambda^\top(y-z):$$
$$0\geq\beta^\top y+\min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\Big\}\qquad\geq$$
$$\max_{\lambda\in E_o}\ \min_{y\in\mathbb{R}^n_+,(x,z)\in conv(\tilde{X})}\Big\{f^\top y+c^\top x+\lambda^\top(y-z):$$
$$0\geq\beta^\top y+\min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\Big\}$$

The last inequality is true given that we have removed the constraint set $By\geq b$. One should notice that the last max-min problem is the Lagrangian dual associated with

$$\min_{y\in\mathbb{R}^n_+,(x,z)\in conv(\tilde{X})}\{f^\top y+c^\top x:\ y=z,\ 0\geq\beta^\top y+\min_{(x,z',v,u)\in conv(\tilde{H})}\{\mathbb{1}^\top v+\mathbb{1}^\top u-\beta^\top z'\}\ \forall\beta\in E_r\},$$

where $y = z$ is priced into the objective function. This problem is equivalent to

$$\min_{(x,z)\in conv(\tilde{X})} \{f^\top z + c^\top x : \ 0 \geq \beta^\top z + \min_{(x,z',v,u)\in conv(\tilde{H})} \{\mathbb{1}^\top v + \mathbb{1}^\top u - \beta^\top z'\} \ \forall \beta \in E_r\}.$$

Recalling that $conv(\tilde{X}) = X \cap \{z \in \mathbb{Z}^n_+\}$, by expanding $conv(\tilde{X})$, we rewrite the previous minimization problem as

$$\min_{(x,z)\in \mathbb{R}^m_+ \times \mathbb{Z}^n_+} \Big\{ f^\top z + c^\top x : \ Bz \geq b, \ Wx + Tz \geq h, \\ 0 \geq \beta^\top z + \min_{(x,z',v,u)\in conv(\tilde{H})} \{\mathbb{1}^\top v + \mathbb{1}^\top u - \beta^\top z'\} \ \forall \beta \in E_r \Big\}.$$

In the above formulation the feasibility cuts become redundant since any z is guaranteed to satisfy $Wx + Tz \geq h$. We thus obtain

$$\min_{(x,z)\in \mathbb{R}^m_+ \times \mathbb{Z}^n_+} \{f^\top z + c^\top x : \ Bz \geq b, \ Wx + Tz \geq h\}$$

which is equivalent to the original problem (5.1).

APPENDIX H PROOF OF PROPOSITION 5.3

We have $\theta \geq \max_{\lambda \in \mathbb{R}^n} \min_{(x,z)\in X} \{c^\top x - \lambda^\top (z - y^*) : z \in \mathbb{Z}_+^n\} \geq \max_{\lambda \in \mathbb{R}^n} \min_{(x,z)\in X} \{c^\top x - \lambda^\top (z - y^*) : z_I \in \mathbb{Z}_+^{|I|}\} \geq \max_{\lambda \in \mathbb{R}^n} \Big\{ \min_{(x,z)\in X} \{c^\top x - \lambda^\top (z - y^*) : z_I \in \mathbb{Z}_+^{|I|}\} : \lambda_{I'} = \hat{\lambda}_{I'} \Big\} = \max_{\lambda \in \mathbb{R}^n} \Big\{ \min_{(x,z)\in X : z_I \in \mathbb{Z}_+^{|I|}} \{c^\top x - \lambda^\top (z - y^*)\} : \lambda_{I'} = \hat{\lambda}_{I'} \Big\} \geq \max_{\lambda \in \mathbb{R}^n} \Big\{ \min_{(x,z)\in X : z_I \in \mathbb{Z}_+^{|I|}} \{c^\top x - \lambda^\top (z - y^*) : z_{I'} = ub_{I'}\} : \lambda_{I'} = \hat{\lambda}_{I'} \Big\}$. Note that the last inequality is true due to our assumption and the condition imposed on $\hat{\lambda}_{I'}$.

APPENDIX I PROOF OF PROPOSITION 5.4

Let us assume that $(\bar{x}, \bar{z})$ is the actual optimal solution of the subproblem and let $Q(\bar{x}, \bar{z})$ and $Q(\hat{x}(\epsilon), \hat{z}(\epsilon))$ be the objective value of the minimization problem for the optimal and ϵ-optimal solutions, respectively. Since the solution $(\bar{x}(\epsilon), \bar{z}(\epsilon))$ is ϵ-optimal, we have $Q(\hat{x}(\epsilon), \hat{z}(\epsilon)) - Q(\bar{x}, \bar{z}) \leq \epsilon$ or equivalently $Q(\bar{x}, \bar{z}) \geq Q(\hat{x}(\epsilon), \hat{z}(\epsilon)) - \epsilon$. We know that $\theta \geq Q(\bar{x}, \bar{z}) + y^\top \lambda^*$ for all $y \in Y$ and thus $\theta \geq Q(\hat{x}(\epsilon), \hat{z}(\epsilon)) - \epsilon + y^\top \lambda^* = c^\top \hat{x}(\epsilon) + (y - \hat{z}(\epsilon))^\top \lambda^* - \epsilon$ for all $y \in Y$.

APPENDIX J LAGRANGIAN DUAL DECOMPOSITION METHOD

Without lose of generality, we rewrite problem (5.1) as follows

$$\min_{y,x,z}\{f^\top y + c^\top x :\ By \geq b,\ Wx + Tz \geq h,\ z = y,\ \ y \in \mathbb{Z}^n_+,\ z \in \mathbb{Z}^n_+,\ x \in \mathbb{R}^m_+\}$$

Pricing out the equality constraint $z = y$ into the objective function using dual multiplier $\lambda \in \mathbb{R}^n$ results into following Lagrangian dual problem

$$\max_{\lambda\in\mathbb{R}^n}\ \min_{y,x,z}\{f^\top y + c^\top x + \lambda^\top(y - z) :\ By \geq b,\ Wx + Tz \geq h,\ \ y \in \mathbb{Z}^n_+,\ z \in \mathbb{Z}^n_+,\ x \in \mathbb{R}^m_+\} =$$
$$\max_{\lambda\in\mathbb{R}^n}\ \min_{y,x,z}\{(f + \lambda)^\top y + c^\top x - \lambda^\top z :\ By \geq b,\ Wx + Tz \geq h,\ \ y \in \mathbb{Z}^n_+,\ z \in \mathbb{Z}^n_+,\ x \in \mathbb{R}^m_+\}$$

for which the minimization problem can be optimized separately over y and (z, x) variables.

APPENDIX K EXAMPLE

In this part, we provide the details of solving the toy example (5.15) when the strengthened Benders cuts and the Lagrangian cuts are used.

Strengthened Benders cuts

Considering the binary variable as the complicating one, we derive the following relaxed LP master problem,

$$\min_{y,\theta}\{\theta : 1 \geq y \geq 0\}, \tag{A.1}$$

and the following subproblem

$$\begin{aligned}\min_{z,x}\Big\{x : x + 15z \geq 8,\ 3x + 10z \geq 13,\ x + 10z \geq 7,\\ 2x - 10z \geq -1,\ 2x - 70z \geq -49,\ z = \bar{y},\ z \in [0,1]\Big\}\end{aligned} \tag{A.2}$$

where z is a copy of the y variable and $\bar{y}$ is the current master solution. Solving the master problem (A.1) yields $\bar{y} = 0$. This is an integer point and thus we only generate the (generalized) Benders cut by solving subproblem (A.2) for $\bar{y} = 0$. This gives $x = 8$ and $\lambda = -15$, where λ is the multiplier associated to the constraint $z - \bar{y} = 0$. We can thus generate an optimality cut and updates the master problem (A.1) as follows :

$$\min_{y,\theta}\{\theta :\ \theta \geq 8 - 15y,\ 1 \geq y \geq 0\} \tag{A.3}$$

In next iteration, solving the master problem (A.3) gives $y = 1$ and solving the subproblem (A.2) for $\bar{y} = 1$ yields $x = 10.5$ and $\lambda = 35$. Thus, we can generate a new optimality cut and update the master problem as follows :

$$\min_{y,\theta}\{\theta : \theta \geq 8 - 15y,\ \theta \geq -\frac{49}{2} + 35y,\ 1 \geq y \geq 0\} \tag{A.4}$$

Solving the above master problem we get a lower bound of -1.75 and $y = 0.65$. We get $\lambda = 5$ by solving subproblem (A.2) for $\bar{y} = 0.65$, resulting the Benders cut $\theta \geq -0.5 + 5y$. Since the master solution is fractional, we can generate the strengthened Benders cut by solving the following Lagrangian dual subproblem (obtained from relaxing $z = \bar{y}$ into objective function

and imposing integrality requirement on the z variable) :

$$\min_{z,x} \Big\{ x + 5(\bar{y} - z) : x + 15z \geq 8,\ 3x + 10z \geq 13,\ x + 10z \geq 7,\\ 2x - 10z \geq -1,\ 2x - 70z \geq -49,\ z \in \{0,1\} \Big\} \tag{A.5}$$

the optimal solution of the above problem is $z = 1$ with objective value of 5.5. This gives the strengthened Benders cuts $\theta \geq 5.5 + 5y$ which is 6 units tighter than the classical Benders at any master solution. Adding this cut to the master problem (A.4), we get

$$\min_{y,\theta} \{ \theta : \theta \geq 8 - 15y,\ \theta \geq -\frac{49}{2} + 35y,\ \theta \geq 5.5 + 5y,\ 1 \geq y \geq 0 \} \tag{A.6}$$

with $y = 0.125$ and the lower bound of 6.125. Executing the next iteration, we observe that this is the best solution that we can get for the LP relaxation of the master problem.

Lagrangian cuts

Note from section K that the master problem generates integer solutions for the first two iterations. As we mentioned earlier, the classical Benders cuts are the tightest at the integer points. We thus start directly from following master problem to avoid reduplication of the results,

$$\min_{y,\theta} \{ \theta : \theta \geq 8 - 15y,\ \theta \geq -\frac{49}{2} + 35y,\ 1 \geq y \geq 0 \} \tag{A.7}$$

which gives the lower bound of -1.75 and $y = 0.65$. To generate the Lagrangian cut, we need to solve the following Lagrangian dual problem :

$$\max_{\lambda} \Big\{ \lambda\bar{y} + \min_{z,x} \{ x - \lambda z : x + 15z \geq 8, 3x + 10z \geq 13, x + 10z \geq 7,\\ 2x - 10z \geq -1, 2x - 70z \geq -49, z \in \{0,1\} \} \Big\} \tag{A.8}$$

To solve (A.8), we use the subgradient method. To initiate the λ value, we solve problem (A.2) with $\bar{y} = 0.65$ which gives $\lambda = 5$. For this λ value, we solve the inner minimization problem of (A.8), i.e.,

$$\min_{z,x} \Big\{ x - 5z : x + 15z \geq 8,\ 3x + 10z \geq 13,\ x + 10z \geq 7,\\ 2x - 10z \geq -1,\ 2x - 70z \geq -49,\ z \in \{0,1\} \Big\}, \tag{A.9}$$

which gives $z = 1$ and $x = 5.5$. Notice that $z = 1$ is a feasible solution to the original problem and its associated cost is 10.5 units. We can thus update the upper bound at this step. Given the lower bound of -1.75, upper bound of 10.5, a step size of $\frac{10}{49}$, $\bar{y} = 0.65$, and $z = 1$, we update the dual multiplier as follows $\lambda = 5 - \frac{10}{49}\frac{(10.5+1.75)}{||1-0.65||_2^2}(1 - 0.65) = 2.5$. We solve again the

Lagrangian subproblem (A.9) with the new λ multiplier, i.e.,

$$\min_{z,x} \Big\{ x - 2.5z : x + 15z \geq 8, 3x + 10z \geq 13, x + 10z \geq 7, \\ 2x - 10z \geq -1, 2x - 70z \geq -49, z \in \{0, 1\} \Big\}. \tag{A.10}$$

This gives $x = 8$, $z = 0$, and an optimality cut of $\theta \geq 8 + 2.5y$. It is easy to observe that this cut gives the convex hull representation of the master problem. We thus terminate the subgradient method. Moreover, $z = 0$ is a new integer solution which is feasible to the original problem, we can thus update the upper bound value from 10.5 to 8.

APPENDIX L THE TEST PROBLEMS

In this section, we give the mathematical formulation of our test problems.

Multicommodity capacitated network design

This problem is defined on a directed graph with a set of potential arcs A and the node set N. The goal is to chose the most appropriate subset of the arcs such that each commodity k, from set K, can flow from its unique origin node $O(k) \in N$ to its unique destination node $D(k) \in N$, while the total cost is minimum. Three parameters are associated to each arc $a \in A$, i.e., fixed cost f_a, capacity u_a and flow cost c_a^k per unit of flow for commodity $k \in K$. Each commodity $k \in K$ is associated with a stochastic demand characterized by d_s^k for each observation of the uncertainty $s \in S$. To model this problem, we define binary variables y_a (equal to 1 if arc $a \in A$ is chosen and 0 otherwise) and continuous variables $x_s^{a,k}$ (to measure the amount of flow on arc $a \in A$ under observation $s \in S$ for commodity $k \in K$). Thus, the extensive formulation of this problem is

$$ND := \min_{y \in \{0,1\}^{|A|},\ x \in \mathbb{R}_+^{|A|\times|K|\times|S|}} \sum_{a \in A} f_a y_a + \sum_{s \in S} \sum_{a \in A} \sum_{k \in K} p_s c_a^k x_s^{a,k} \tag{B.1}$$

$$\text{s.t.} \sum_{a \in A^+(i)} x_s^{a,k} - \sum_{a \in A^-(i)} x_s^{a,k} = d_s^i \qquad \forall i \in N, k \in K, s \in S \tag{B.2}$$

$$\sum_{k \in K} d_s^k x_s^{a,k} \leq u_a y_a \qquad \forall a \in A, s \in S \tag{B.3}$$

where $A^+(i)/A^-(i)$ indicates the set of outward/inward arcs to node $i \in N$, p_s is probability of scenario s and d_s^i is equal to $d_s^k \left(-d_s^k\right)$ if $i = O(k)$ $\left(D(k)\right)$, and 0 otherwise.

Stochastic capacitated facility location problem

To define this problem, let N be the set of potential locations for the facilities and M the set of customers. The goal is to open sufficient facilities to satisfy all the demand at minimum cost. Each customer may be served by one or several facilities. At each potential location $i \in N$, at most one facility with service capacity of u_i can be opened which entails a fixed cost of f_i units. The routing cost from customer $j \in M$ to facility $i \in N$ per unit of flow is c_{ij}. Each customer $j \in M$ has a stochastic demand d_j^s, where $s \in S$ is a specific realization of the uncertainty with probability of p_s such that $\sum_{s \in S} p_s = 1$.

Let binary variable y_i take 1 if a facility is opened at location $i \in N$ and 0 otherwise, and let

$x_{ij}^s \geq 0$ indicate the amount of flow from customer $j \in J$ to facility $i \in I$ under realization $s \in S$. We thus use following formulation of this problem

$$SFL := \min_{y \in \{0,1\}^{|N|},\ x \in \mathbb{R}_+^{|N|\times|M|\times|S|}} \sum_{i \in N} f_i y_i + \sum_{s \in S} \sum_{i \in N} \sum_{j \in M} p_s c_{ij} x_{ij}^s \tag{B.4}$$

$$\text{s.t.} \sum_{i \in N} x_{ij}^s \geq d_j^s \qquad \forall j \in M, s \in S \tag{B.5}$$

$$\sum_{j \in M} x_{ij}^s \leq u_i y_i \qquad \forall i \in N, s \in S \tag{B.6}$$

$$\sum_{i \in N} u_i y_i \geq \max_{s \in S} \sum_{j \in M} d_j^s \tag{B.7}$$

The objective function minimizes the total fixed costs of opening facility plus the expected flow cots. First constraint set imposes the demand satisfaction for each customer in every scenario. Second constraint set imposes the capacity restriction on each facility and the last constraint is to add the relatively complete recourse property to the problem.

Stochastic network interdiction

This problem is defined over a directed graph consisting a set of nodes N, arcs A, and a subset of candidate arcs $L \subseteq A$ on which sensors can be installed. The goal is to maximize the probability of catching an intruder that traverses some path in the network. The first-stage decisions determine whether or not to install a sensor on arc $a \in L$ knowing the probability of the intruder avoiding detection with and without a sensor on this arc (denoted by r_a and q_a). A scenario $s \in S$, with probability of p_s, corresponds to the origin O_s and destination D_s of the intruder. Cost of installing of a sensor on arc $a \in D$ is c_a units and there is a total budget of b units to install sensors. In the second-stage the intruder choses a maximum-reliability path from its origin to the targeted node that maximizes the probability of avoiding detection. The maximum-reliability path from node $j \in N$ to destination D_s is represented by ψ_s^j.

The first-stage binary variables y_a takes value of 1 if a sensor is installed on arc $a \in L$ and 0 otherwise. The second-stage variables $\pi_s^i,\ \forall i \in N, s \in S$ model the probability of reaching destination D_s from node i without being detected. The formulation of this problem is as

follows :

$$SNI := \min_{y\in\{0,1\}^{|L|},\pi\in\Re_{+}^{|N||S|}} \sum_{s\in S} p_s \pi_s^{O_s} \tag{B.8}$$

$$\text{s.t.} \sum_{a\in L} c_a y_a \leq b \tag{B.9}$$

$$\pi_s^{D_s} = 1 \qquad \forall s \in S \tag{B.10}$$

$$\pi_s^{a^+} - q_a \pi_s^{a^-} \geq 0 \qquad \forall a \in L, s \in S \tag{B.11}$$

$$\pi_s^{a^+} - r_a \pi_s^{a^-} \geq 0 \qquad \forall a \in A\backslash L, s \in S \tag{B.12}$$

$$\pi_s^{a^+} - r_a \pi_s^{a^-} \geq -(r_a - q_a)\psi_s^{a^-} y_a \quad \forall a \in L, s \in S \tag{B.13}$$

where a^+ and a^- indicate the tail and head of arc a. Constraint (B.9) imposes the budget restriction. The remaining constraints calculate the least probability that the intruder can reach to the destination undetected.

APPENDIX M PROOF OF PROPOSITION 6.1

It is trivial to observe that $\lceil \bar{y} \rceil \in Y$ due to assumption (ii). Let assume that the recourse cost associated to the rounded solution $\lceil \bar{y} \rceil$ is known and given by ν'_s for each $s \in \mathcal{S}$. Thus, for this integer solution, a valid upper bound can be calculated from $f^\top \lceil \bar{y} \rceil + \sum_{s \in \mathcal{S}} \rho_s \nu'_s \geq z^*$. On the other hand, based on the assumptions (ii) and (iii), we have $\nu^*_s = Q(\bar{y}, s) \geq Q(\lceil \bar{y} \rceil, s) = \nu'_s$. Thus, $f^\top \lceil \bar{y} \rceil + \sum_{s \in \mathcal{S}} \rho_s \nu^*_s \geq f^\top \lceil \bar{y} \rceil + \sum_{s \in \mathcal{S}} \rho_s \nu'_s \geq z^*$. $\square$

APPENDIX N PROOF OF PROPOSITION 6.2

If $|\mathcal{S}_g| = 1$, the results follows immediately from the disaggregated version of the BD method. To proof the validity of the cuts for $1 < |\mathcal{S}_g| \leq |\mathcal{S}|$, we need to show that the derived SP gives a valid lower approximation of the aggregated recourse variables for any $y \in Y$.

$$\theta_s \geq \min_{x \in \mathbb{R}^m_+} \{c^\top x_s : \ Wx_s \geq h_s - T_s y\} \qquad s \in \mathcal{S}_g$$

$$\beta_s \geq 0 \ \rightarrow \ \beta_s \theta_s \geq \min_{x \in \mathbb{R}^m_+} \{\beta_s c^\top x_s : \ Wx_s \geq h_s - T_s y\} \qquad s \in \mathcal{S}_g$$

$$\sum_{s \in \mathcal{S}_g} \beta_s \theta_s \geq \min_{x \in \mathbb{R}^{m|\mathcal{S}|}_+} \{ \sum_{s \in \mathcal{S}_g} \beta_s c^\top x_s : Wx_s \geq h_s - T_s y, \ s \in \mathcal{S}_g\} \geq \min_{x \in \mathbb{R}^{m|\mathcal{S}|}_+} \Big\{ c^\top \sum_{s \in \mathcal{S}_g} \beta_s x_s : \sum_{s \in \mathcal{S}_g} W \beta_s x_s \geq \sum_{s \in \mathcal{S}_g} \beta_s (h_s - T_s y) \Big\}$$

The last inequality holds because we have aggregated the constraints using convex combination weights $1 \geq \beta_s \geq 0$ such that $\sum_{s \in \mathcal{S}_g} \beta_s = 1$. We next consider a variable transformation $\sum_{s \in \mathcal{S}_g} \beta_s = 1, \ x_g = \sum_{s \in \mathcal{S}_g} \beta_s x_s, \ h_g = \sum_{s \in \mathcal{S}_g} \beta_s h_s, \ T_g = \sum_{s \in \mathcal{S}_g} \beta_s T_s$. Thus,

$$\sum_{s \in \mathcal{S}_g} \beta_s \theta_s \geq \min_{x \in \mathbb{R}^m_+} \{c^\top x_g : \ Wx_g \geq h_g - T_g y\} = \max_{\alpha \in \mathbb{R}^l_+} \{(h_g - T_g y)^\top \alpha : \ W^\top \alpha \leq c\} \quad \forall y \in Y,$$

Also, we observe that the dual polyhedron is identical to a regular dual SP which indicates the validity of the feasibility cut. $\square$

APPENDIX O PROOF OF THEOREM 6.1

For an arbitrary first-stage solution $y \in Y$, let $\sigma^g = (h_g - T_g y)^\top \alpha_g$ be the right hand side of the optimality cut generated from artificial scenario $g \in \mathcal{G}$ and let $\theta_s^* = (h_s - T_s y)^\top \alpha_s^{i^*}$, where $i^* \in \arg\max_{i \in E_s^t} (h_s - T_s y)^\top \alpha_s^i$ for which E_s^t is the set of optimality cuts associated to scenario s at iteration t. If the cut from the artificial SP is violated by the y solution, it means that $\sigma^g > \sum_{s \in \mathcal{S}_g} \beta_s \theta_s^*$. For a given y solution, the MP can be separated for each cluster $g \in \mathcal{G}$ and can be defined as :

$$\begin{aligned} \min \sum_{s \in \mathcal{S}_g} \rho_s \theta_s \\ \sum_{s \in \mathcal{S}_g} \beta_s \theta_s \geq \sigma^g \\ \theta_s \geq \theta_s^* \qquad s \in \mathcal{S}_g. \end{aligned}$$

It can be shown that the above formulation has a knapsack structure and since $\sigma^g > \sum_{s \in \mathcal{S}_g} \beta_s \theta_s^*$, the recourse variable $\theta_{\tilde{s}}$ takes the extra violation $\sigma^g - \sum_{s \in \mathcal{S}_g} \beta_s \theta_s^* > 0$, where $\tilde{s} \in \arg\min_{s \in \mathcal{S}_g} \frac{\rho_s}{\beta_s}$. This gives $\theta_{\tilde{s}} = \theta_{\tilde{s}}^* + \frac{1}{\beta_{\tilde{s}}}(\sigma^g - \sum_{s \in \mathcal{S}_g} \beta_s \theta_s^*)$ which results in a lower bound improvement of $\Delta := \frac{\rho_{\tilde{s}}}{\beta_{\tilde{s}}}(\sigma^g - \sum_{s \in \mathcal{S}_g} \beta_s \theta_s^*)$ at the given y solution. The maximum value of Δ is achieved when $\beta_s = \rho_s$. Considering $\sum_{s \in \mathcal{S}_g} \beta_s = 1$ and $\mathcal{S}_\} \leq |\mathcal{S}|$, the maximum value of Δ is achieve at $\beta_s = \frac{\rho_s}{\sum_{s \in \mathcal{S}_g} \rho_s}$. $\square$

APPENDIX P STOCHASTIC NETWORK DESIGN PROBLEM

The MCFNDSD is defined on a directed graph consisting a set of nodes $\mathcal{N}$ and a set of potential arcs $\mathcal{A}$. In this problem, a set of commodities $\mathcal{K}$ exist where each commodity $k \in \mathcal{K}$ has an uncertain amount of demand that needs to be routed from its unique origin $O(k) \in \mathcal{N}$ to its unique destination $D(k) \in \mathcal{N}$. We assume that the demand is characterized with a set of discrete scenarios $\mathcal{S}$. Therefore, each commodity $k \in \mathcal{K}$ has a stochastic demand which is denoted by $d_s^k \geq 0$ for each possible scenario $s \in \mathcal{S}$. We assume that the realization probability for each scenario $s \in \mathcal{S}$ is known and denoted by ρ_s such that $\sum_{s \in \mathcal{S}} \rho_s = 1$. The goal is to select a proper subset of the arcs to meet all the flow requirements at minimum cost. To use arc $a \in \mathcal{A}$ we need to pay a fixed cost of f_a units and to route a unit of commodity $k \in \mathcal{K}$ on this arc c_a^k units will be charged. In addition, there is a capacity limit u_a on each arc $a \in \mathcal{A}$. Thus, the objective function is to minimize sum of the fixed costs and the expected flow costs.

To model this stochastic problem we define the binary first-stage variables y_a indicating if arc $a \in \mathcal{A}$ is used 1 or not 0. To model the second-stage, we define continuous variables $x_a^{k,s} \geq 0$ to reflect the amount of flow on arc $a \in \mathcal{A}$ for commodity $k \in \mathcal{K}$ under realization $s \in \mathcal{S}$. The extensive formulation of MCFNDSD is thus :

$$MCFNDSD = \min_{y \in \{0,1\}^{|\mathcal{A}|}, x \in \Re_+^{|\mathcal{A}||\mathcal{K}||\mathcal{S}|}} \sum_{a \in \mathcal{A}} f_a y_a + \sum_{s \in \mathcal{S}} \rho_s \sum_{k \in \mathcal{K}} \sum_{a \in A} c_a^k x_a^{k,s} \tag{B.1}$$

$$\text{s.t} : \sum_{a \in \mathcal{A}(i)^+} x_a^{k,s} - \sum_{a \in \mathcal{A}(i)^-} x_a^{k,s} = \begin{cases} d_s^k & \text{if } i = O(k) \\ -d_s^k & \text{if } i = D(k) \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K}, s \in \mathcal{S} \tag{B.2}$$

$$\sum_{k \in \mathcal{K}} x_a^{k,s} \leq u_a y_a \qquad \forall a \in \mathcal{A}, s \in \mathcal{S}, \tag{B.3}$$

where $\mathcal{A}(i)^+$ and $\mathcal{A}(i)^-$ indicate the set of outward and inward arcs incident to node i. The objective function minimizes the total fixed costs plus the expected routing costs. For each scenario, constraint set (B.2) imposes the flow conservation requirements for each commodity and node. Constraints (B.3) enforce the capacity limit on each arc in every scenario. To introduce the complete recourse property for the above formulation, we add a dummy arc between each O-D pair with large routing cost as an outsourcing strategy.

APPENDIX Q TEST INSTANCES

Table Q.1 presents the detail of the used instances problems from **R** family which we used in this article.

Table Q.1 Attributes of the instance classes

Name	$\|N\|$	$\|A\|$	$\|K\|$	$\|\Omega\|$	Cost/Capacity Ratio	Correlation	#Instances
r04	10	60	10	1000	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	25
r05	10	60	25	1000	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	25
r06	10	60	50	1000	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	25
r07	10	82	10	1000	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	25
r08	10	83	25	1000	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	25
r09	10	83	50	1000	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	25
r10	20	120	40	1000	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	25
r11	20	120	100	1000	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	25

APPENDIX R NUMERICAL RESULTS OF THE SEQUENTIAL ALGORITHM

To make fair comparisons, we have incorporated the techniques that we developed in sections 6.4.1, 6.4.2, and 6.4.4 into our sequential method (presented in Algorithm 2). We have also incorporated the classical acceleration techniques which we have used in our parallel algorithms, see section 6.7.

In this appendix, we present some numerical results to complement our numerical assessments in section 6.8. In Figure R.1, we thus study impact of the cut aggregation over the sequential algorithm. For each aggregation level, we have ran the algorithm for 2 hours. The value on each column indicates the average optimality gap in percentages.

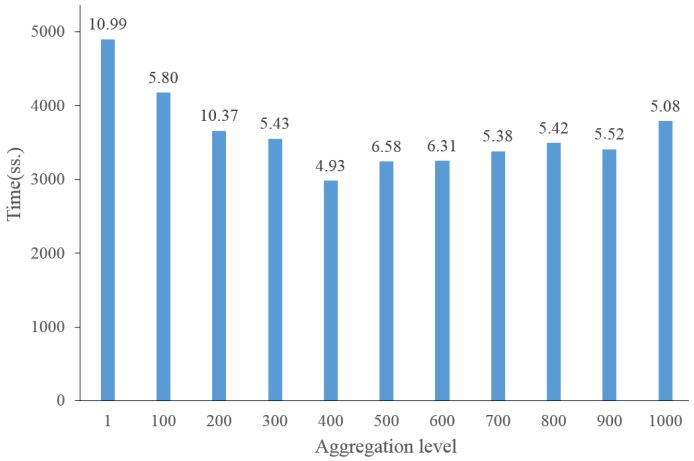

Figure R.1 Comparison of various cut aggregation levels for the sequential BD method

We next present the numerical performance of the sequential algorithm for a run time limit of 10 hours. The results are presented in Table R.1.

Table R.1 Numerical results of the sequential BD algorithm

	#Instance	Time to solve LP	Total Time	Gap(%)	Sol.(%)
r04	3	31.43	1098.21	0.36	100.00
r05	3	159.20	798.77	0.47	100.00
r06	3	775.99	14671.58	1.23	66.67
r07	3	34.81	12087.66	0.76	66.67
r08	3	224.44	12259.00	2.34	66.67
r09	3	1188.19	24270.19	2.79	33.33
r10	3	3582.01	24521.04	6.80	33.33
Ave.	**3**	**856.58**	**12815.21**	**2.11**	**66.67**